Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

ESTUDIO, IMPLEMENTACIÓN Y ANÁLISIS DE MÉTODOS DE TRILATERACIÓN PARA LA LOCALIZACIÓN DE USUARIOS DESDE SUS TERMINALES MÓVILES

Autora: Débora Gómez Bertoli

Tutora: Alicia Rodríguez Carrión

Leganés, 26 de julio de 2012

TÍTULO: Estudio, implementación y análisis de métodos de trilateración para la localización de usuarios desde sus terminales móviles
AUTORA: Débora Gómez Bertoli
TUTORA: Alicia Rodríguez Carrión

EL TRIBUNAL

Presidente: Carlos García Rubio

Vocal: Carlos Jesús Bernardos Cano

Secretaria: Amaya Jiménez Moreno

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 25 de Julio de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Cuando uno llega a este momento en el que la entrega de un proyecto de fin de carrera no simboliza solo el hecho de obtener un título como ingeniero, sino también marca el final de una etapa que en la que mirando hacia atrás te das cuenta de los malos y de los buenos momentos que engloba y que han influido en convertirte lo que eres a día de hoy, lo único en lo que se puede pensar es en toda la gente que te ha rodeado durante este período de tu vida.

Pues este logro no solo es mérito propio si no también de todas las personas que están a tu lado y que a su modo han aportado su granito de arena para llegar hasta donde he llegado. Por ello, no puedo evitar dedicar esta página a todos aquellos que me han apoyado, y a los que solo me sale dedicarles una simple palabra, GRACIAS.

GRACIAS a mis compañeros de carrera que en momentos críticos me ha ayudado a salir adelante, y que tantos momentos, que podrían llegar a clasificarse como estúpidos, han servido como referencia para disfrutar y aprovechar lo que esta etapa de mi vida me ha ofrecido.

GRACIAS a mis carlinhos por todos los momentos que hemos pasado juntos y por haberme permitido ser miembro de esta grandiosa asociación, BEST, de la que me siento orgullosa día a día viendo como se hace cada vez más importante.

GRACIAS a mi tutora y sus tutores, Carlos y Celeste, por haberme tendido la mano siempre que he necesitado ayuda. Vuestra disponibilidad y los buenos momentos que hemos tenido en nuestras reuniones tienen gran valor para mí.

GRACIAS a mi familia por haber hecho posible, que tras finalizar mi ingeniería técnica, haya podido estudiar una ingeniería superior, y por darme consejos y apoyo tanto en los momentos difíciles como en los buenos. Sin vosotros esto no habría sido posible.

GRACIAS a ti por haber estado a mi lado todos estos años, por haber sido mi punto de apoyo. Sin ti no sería la persona que soy ahora.

Resumen

Los teléfonos inteligentes, o *Smartphones*, con acceso a Internet y receptores GPS son los que actualmente dominan el mercado de telefonía móvil puesto que permiten instalar una amplia variedad de aplicaciones de terceros disponibles en lo que se conoce como *App Stores*.

En los últimos años las aplicaciones basadas en localización y posicionamiento (Location-based Services, LBS) han emergido y componen una gran parte del mercado de aplicaciones móviles.

El marco de este proyecto comprende un estudio entre las distintas técnicas de localización y posicionamiento más adecuadas para los dispositivos móviles, más concretamente, se analizarán las competencias en este ámbito de los teléfonos con Android como sistema operativo.

Concretamente, se van a estudiar algoritmos de trilateración que nos permitirán calcular la posición del usuario partiendo de la posición de las tres estaciones base de las cuales recibimos mayor nivel de potencia. El objetivo que nos proponemos haciendo uso de esta técnica de posicionamiento es localizar al usuario con el menor coste de recursos posible.

Para ello, el proyecto consta de una primera fase de desarrollo de una aplicación de posicionamiento para dispositivos Android. Más adelante se tomarán medidas de las posiciones calculadas para el usuario respecto a la posición obtenida mediante un dispositivo GPS autónomo que nos permitirá, posteriormente, evaluar la precisión, el consumo de batería, la latencia y el coste de computación de la aplicación desarrollada.

Este análisis de prestaciones confirma la teoría de que el uso de GPS en los dispositivos móviles consume más energía que el posicionamiento mediante algoritmos de trilateración. La contrapartida es que la precisión es menor en el segundo caso. Además, se ha descubierto que la latencia para el posicionamiento mediante trilateración se ve incrementada debido a las comunicaciones HTTP que se emplean para obtener las posiciones de las estaciones base, mientras que el tiempo de cómputo de los algoritmos de trilateración se puede considerar despreciable.

Abstract

The Mobile phones market is being dominated by Smartphones with Internet access and GPS receivers since they allow running a wide array of third party applications available in special App Stores.

Location-based applications have emerged during the last few years, applications and they compose a wide part within the mobile applications market.

The goal of this project is to study the different location-based techniques more adequate for mobile devices; especially the competences of the Android operating system devices will be analyzed.

Precisely, we are going to study trilateration algorithms that will allow us to obtain the user localization using the coordinates of the three closest base stations with the highest power level received. The main goal of using this positioning technique is to locate the user with a lower cost of resources.

This project is composed by a development phase of a positioning application for Android mobile pohnes. Later, some measurements of the user position will be taken and they will be compared with the positions obtained by a GPS integrated device which will allow us to evaluate the precision, the power consumption, the latency and the computation cost of the developed application.

This analysis about the application benefits confirms the theory of the GPS having greater energy consumption than the positioning algorithms based on trilateration techniques. The counterpart is that the precision is better in the first case. Moreover, it has been identified that the latency in the second case has been increased by the HTTP communications in order to obtain the locations of the base stations while the trilateration algorithms computation time can be considered worthless.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

Este capítulo expone la motivación de este proyecto, sus distintas fases de desarrollo e indica la estructura de esta memoria.

1.1. Motivación del proyecto

Actualmente, cada vez existen más aplicaciones que hacen un uso comercial de la posición del usuario. La explotación de este tipo de información no se trata de un hecho aislado, sino que desde el principio de los tiempos las personas han necesitado conocer su posición para poder situarse y comunicarse con otras personas.

Si nos remontamos a las tribus americanas o a los ancestros chinos, ya se utilizaban señales de humo no sólo para marcar la localización de sus casas o poblados sino también para comunicarse enviando mensajes a otras civilizaciones. Además, durante miles de años los navegantes han estado utilizando las matemáticas para determinar sus coordenadas midiendo el ángulo respecto al sol o a las estrellas. A lo largo de bastantes siglos, no existía una forma fiable de determinar la longitud del océano, y a mediados del siglo XVIII, un relojero llamado John Harrison inventó un cronómetro que permitía a los navegantes determinar la longitud haciendo un seguimiento del cambio del tiempo entre su tierra y su actual posición.

Más tarde, se inventó la brújula magnética que empleando un imán que apuntase a los polos de la Tierra, permitía a los navegantes determinar su dirección en términos de longitud y latitud. A principios del siglo XX, midiendo el nivel de las señales de radio de los barcos, aviones y tropas militares, permitía estimar las coordenadas de su posición a distancias muy lejanas. En este mismo siglo, un conjunto de 30 satélites orbitaban alrededor de la Tierra y se usaban para triangular la posición de un receptor. Estos satélites envían

mensajes con un marca de tiempo, y los receptores calculan la distancia a cada uno de los satélites basándose en la velocidad de los mensajes recibidos comparado con el tiempo en el que fueron enviados.

En los años noventa, los dispositivos de navegación GPS se convirtieron en el primer terminal dedicado exclusivamente a dicha función. Aunque en un principio no tenían mapas y eran lentos, ahora se trata de un dispositivo de navegación muy avanzado que incluye incluso puntos de interés que se muestran en su pantalla conforme se avanza por la ruta que se le ha especificado.

Entrando en el siglo XXI, los teléfonos inteligentes o *smartphones* ya incluían un receptor GPS, sin embargo, la aparición del dispositivo móvil, iPhone, por parte de Apple, introdujo múltiples cambios en este mercado permitiendo instalar aplicaciones de terceros que hiciesen uso del receptor GPS que contenía.

Estos últimos años gracias a la tecnología existente se han desarrollado gran cantidad de servicios basados en localización, juegos que hacen uso de la posición del usuario para compartirla con otros usuarios. Al igual que ocurre con las redes sociales.

Los servicios basados en localización (Location Based Services, LBS) son servicios que utilizan la información de localización del usuario que se obtiene mediante tecnologías de detección para mejorar la relevancia y el contexto de las aplicaciones.

El requisito fundamental de estos servicios es obtener las coordenadas de una localización con la mayor precisión posible, y para ello existen diferentes técnicas que serán presentadas en el siguiente apartado.

De acuerdo con [24], existen tres conceptos básicos dentro del ámbito de los servicios basados en localización:

- Posición. Aparece en forma de coordenadas espaciales y se puede representar con un único punto en coordenadas cartesianas.
- Localización. Asocia la posición con un sitio concreto en el mundo real.
- Servicio de localización. Se distingue de los LBS en que exclusivamente se encarga de la localización del usuario y la pone a disposición de agentes externos.

Algunas de las aplicaciones que se derivan de la utilización de estos servicios se pueden resumir de la siguiente forma y quedan recogidas en la figura 1.1:

- Servicios de emergencia: asistencia en carretera y llamadas al 112.

- Servicios de seguimiento: localizador de familiares y de asistentes de personal.
- Navegación: asistencia, cálculo de rutas y direcciones.
- Facturación: peajes de carretera y aparcamientos.
- Alertas LBS: promociones y anuncios.
- Redes sociales: localizador de amigos y mensajes instantáneos.
- Aplicaciones de operadores de red: localización basada en facturación del usuario, detección y prevención de fraudes, optimización de redes inalámbricas, negociación de fronteras entre redes celulares.
- Aplicaciones de proveedores de servicios: redes inalámbricas máquina a máquina (Machine-to-Machine,M2M), control remoto de datos y televisión en el móvil.

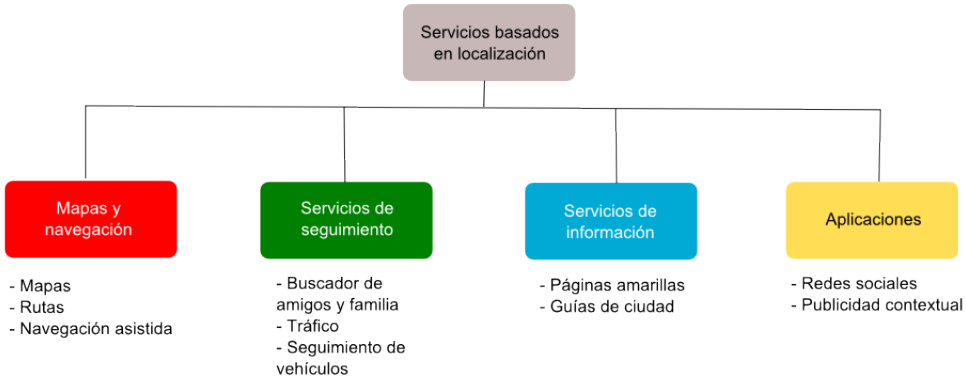


Figura 1.1: Servicios basados en localización

Dado el creciente desarrollo de estos servicios basados en localización, unido al éxito del sistema operativo Android en este mercado, nos proponemos desarrollar una aplicación que nos sirva como herramienta para tomar medidas y evaluar posteriormente algunas de las prestaciones con respecto a las aplicaciones ya existentes en este ámbito, tales como, la precisión, el consumo de batería o la latencia. Tanto el consumo de batería como el tiempo de latencia son la debilidad de las aplicaciones basadas en localización puesto que ambos son, en algunos casos, muy elevados. Nuestra aplicación pretende

hacer frente a estos problemas reduciendo el tiempo de latencia ya que mientras el móvil disponga de cobertura móvil podremos conocer la posición de la estación base a la que estamos conectados en todo momento, sin tener que esperar a establecer una conexión con algún satélite, y como consecuencia reducir también el consumo de potencia.

Además, se pretende realizar un análisis de la precisión de los algoritmos de trilateración con respecto a la posición obtenida por un dispositivo GPS autónomo y evaluar si el error cometido es mayor o menor que el error cometido por Google en el cálculo de la posición del usuario.

1.2. Objetivos

La meta principal de este proyecto realizar una evaluación de la calidad de los algoritmos empleados en la aplicación a implementar para obtener la posición del usuario de forma alternativa a como lo hace Google y el GPS. Para ello, se ha centrado el desarrollo en una plataforma de código libre, Android, que actualmente está muy extendida dentro del mercado de la telefonía móvil. Para poder proporcionar una aplicación simple, con una interfaz amigable y con bajo consumo de batería, y que guardase la información necesaria para evaluar las prestaciones de los algoritmos empleados, se han perseguido los siguientes objetivos a lo largo del desarrollo de este proyecto de fin de carrera:

- Investigar las técnicas de posicionamiento existentes. Puesto que nuestro objetivo es crear una aplicación de posicionamiento autónoma, es importante conocer el entorno y las tecnologías existentes en dicho ámbito para elegir la que más se adecue a nuestro objetivo y mejorar aspectos de nuestra aplicación frente al resto de aplicaciones existentes en este marco.

- Investigar las principales características de Android. Antes de ponernos a implementar la aplicación debemos conocer las herramientas que este sistema operativo nos facilita para llevar a cabo nuestro objetivo principal. Con este motivo se han estudiado los diferentes componentes, tanto visuales como funcionales, que nos permiten conocer la posición del usuario y representarla en la pantalla del dispositivo móvil.

- Investigar métodos de trilateración. Cuando ya hemos escogido la tecnología y conocemos las herramientas que nos proporciona Android y sus limitaciones, podemos investigar algoritmos de trilateración para mejorar la precisión de nuestra aplicación.

- Estudiar el entorno de desarrollo de Android. Para poder implementar aplicaciones en este lenguaje, debemos hacer uso de una serie de herramientas bajo una licencia de software libre disponibles al alcance de cualquier desarrollador. También incluye la configuración del entorno para poder obtener los datos de localización del usuario.

- Desarrollar una aplicación de localización para dispositivos Android. Una vez conocemos las herramientas que cumplen nuestro objetivo y tenemos configurado el entorno, podemos ponernos a implementar la aplicación. Además, Android nos provee con un API documentado que nos facilitará el entendimiento de sus herramientas para desarrollar nuestra aplicación de la forma más eficaz y simple posible.

- Realizar una batería de pruebas. El principal objetivo de este apartado es realizar un barrido de datos de nuestra aplicación en distintas localizaciones y entornos para poder analizar las prestaciones de los algoritmos de trilateración empleados para calcular la posición del usuario.

- Análisis de resultados. Se trata de realizar un análisis exhaustivo de los datos obtenidos en la fase de pruebas y sacar conclusiones acerca de las mejoras que propone el uso de algoritmos de trilateración para localizar al usuario en lugar de emplear un dispositivo GPS como hacen muchas de las aplicaciones existentes.

1.3. Medios empleados

En la realización de este proyecto hemos utilizado un dispositivo móvil con sistema operativo Android 2.2. para la toma de medidas de localización y consumo de batería, y en su posterior procesamiento se ha empleado el programa Matlab para la representación de estadísticas de los datos obtenidos. Además, para comparar la fiabilidad de estos datos se ha empleado un dispositivo GPS autónomo como referencia.

También ha implicado el desplazamiento a diferentes zonas (urbanas y rurales) para realizar comparativas de la calidad de cada una de las posiciones calculadas para el usuario.

1.4. Estructura de la memoria

En este apartado se describe brevemente el contenido de cada uno de los capítulos que componen esta memoria y los anexos adicionales incluídos.

En el capítulo 1, Introducción, se expone el motivo de la realización de este proyecto incluyendo una breve descripción de los objetivos que engloba y una visión general del contenido de la memoria.

El capítulo 2, Estado del arte, se presentan los distintos servicios existentes que hacen uso de la posición del usuario, así como las diferentes tecnologías y las técnicas que se aplican para calcular dicha posición. También se incluye una descripción exhaustiva de los métodos de trilateración que se emplearán para mejorar la precisión de la aplicación. Finalmente, se enumeran las características y componentes principales de nuestro interés que posee la plataforma Android, y las diferentes opciones que nos proporciona para llevar a cabo el desarrollo de nuestra aplicación.

En el capítulo 3, Desarrollo de DroidTriangulation, se enumeran las distintas funcionalidades utilizadas para obtener información sobre la posición del usuario, especificando en algunos casos fragmentos de código relevantes, y exponiendo los motivos de elección de cada uno de los componentes de la aplicación.

En el capítulo 4, Entorno de pruebas, se describe los entornos de realización de pruebas para obtener datos de posicionamientos en diferentes ámbitos y se presentan capturas de mapas para su mayor entendimiento.

El capítulo 5, Análisis de resultados, tiene como objetivo presentar una serie de conclusiones a partir de los datos obtenidos en la fase de pruebas y realizar una comparativa con las tecnologías existentes.

En el capítulo 6, Historia del proyecto, se presentan los pormenores que han surgido a lo largo de la realización de este proyecto, mostrando la evolución de su desarrollo en el tiempo y exponiendo un presupuesto aproximado y justificado de su coste.

El capítulo 7, Conclusiones y trabajos futuros, se muestran las posibles líneas futuras de desarrollo partiendo de una comparación entre los objetivos iniciales del proyecto y los objetivos alcanzados.

Finalmente el proyecto incluye como anexos un manual de instalación de la aplicación Android, un manual de usuario y una guía para generar una clave para poder emplear el API de Google Maps en una aplicación Android.

Capítulo 2

Estado del Arte

En este capítulo se presenta una breve introducción acerca de las diferentes técnicas de posicionamiento existentes, bien mediante el uso de satélites, o bien mediante el uso de redes móviles, y las diferentes metodologías empleadas en cada una de ellas que tienen como principal objetivo mejorar la precisión y reducir el consumo de batería de los dispositivos. Además, se presenta una breve descripción del sistema operativo para el cual se ha desarrollado nuestra aplicación.

2.1. Técnicas de posicionamiento

Las técnicas de posicionamiento hacen uso de tres tecnologías principales: los satélites, las redes de telefonía móvil y las estaciones WiFi. A continuación, se realiza una breve descripción las técnicas utilizadas para cada una de las tecnologías anteriores y se presenta una comparativa de todas las técnicas mencionadas.

2.1.1. Posicionamiento mediante satélites

El sistema de posicionamiento mediante satélites más utilizado en la actualidad es el Sistema de Posicionamiento Global o GPS que utiliza 32 satélites equiespaciados para determinar la posición de un terminal móvil, y que de acuerdo con [23], transmiten en dos frecuencias: L1 (uso público) y L2 (uso militar). Sin embargo, la tecnología conocida como Sistema de Posicionamiento Global Asistido (Assisted-GPS, A-GPS) está cobrando cada vez más fuerza en la actualidad. A continuación se describen más detalladamente.

Servicio de posicionamiento global

Como se ha comentado anteriormente, GPS utiliza un conjunto de satélites para posicionar un teléfono móvil, de modo que dicho teléfono móvil, equipado con un receptor GPS, al menos recoge señal de cuatro de estos satélites que están transmitiendo información de tiempo y orbital. Dicha información es utilizada por el terminal móvil para determinar su posición, de modo que procesando dichas señales calcula la posición 3D (latitud, longitud y altitud) en un radio de 10 metros. Además, no hay límite en el número de usuarios que pueden obtener su posición simultáneamente y las señales de GPS son resistentes a interferencias. Todas estas ventajas son las que convierten a GPS en la tecnología de localización más comercializada hasta el momento.

Sin embargo, una de las principales desventajas que tiene esta tecnología es el alto consumo de batería de los terminales móviles junto con el hecho de que dentro de edificios o recintos cerrados apenas se recibe señal, aunque un estudio reciente [13] ha demostrado que en algunos recintos sí es posible obtener la posición mediante GPS. Además, el tiempo necesario para establecer la posición del usuario por primera vez (Time To First Fix, TTFF) tras un largo periodo de apagado puede llegar a ser de 15 minutos, teniendo que recargar la posición de todos los satélites en la órbita. Por último, el hecho de tener que disponer de un receptor GPS para poder utilizar esta tecnología incrementa su coste.

Servicio de posicionamiento global asistido

La idea básica de A-GPS es establecer una red GPS de referencia cuyos receptores siempre tengan visibilidad y puedan operar continuamente, según se comenta en [26]. Esta red de referencia también está conectada con la infraestructura compuesta por redes de telefonía móvil, y continuamente monitoriza el estado de la constelación en tiempo real y provee datos como la posición aproximada del teléfono (o de la estación base), la visibilidad de los satélites, el efemérides y la corrección de reloj, el efecto Doppler, e incluso el código de fase del ruido pseudo-aleatorio de cada satélite para un momento determinado.

Cuando el teléfono móvil hace una petición de su posición, los datos obtenidos de la red de referencia se transmiten al receptor GPS del teléfono para mejorar el TTFF e incrementar la sensibilidad del sensor. Por lo tanto, y de acuerdo con [24], A-GPS mejora las prestaciones de GPS en términos de precisión (menor error en el posicionamiento), rendimiento (mayor tasa de éxito en el cálculo de la posición), TTFF, consumo de batería y coste. Sin embar-

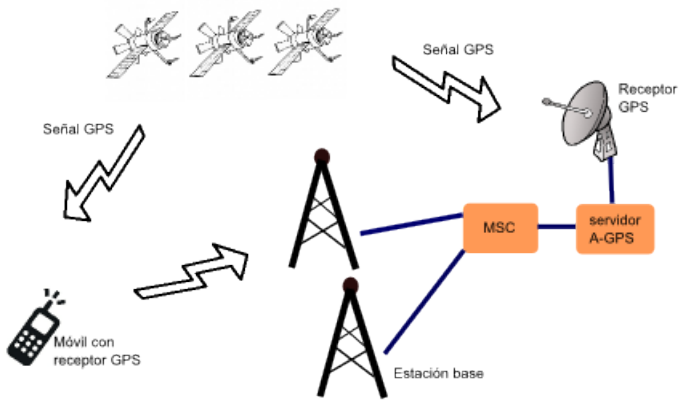


Figura 2.1: Esquema de la tecnología Assisted-GPS

go, se trata de una tecnología más compleja debido a que necesita marcas de tiempo muy precisas y todavía no es un sistema muy comercializado.

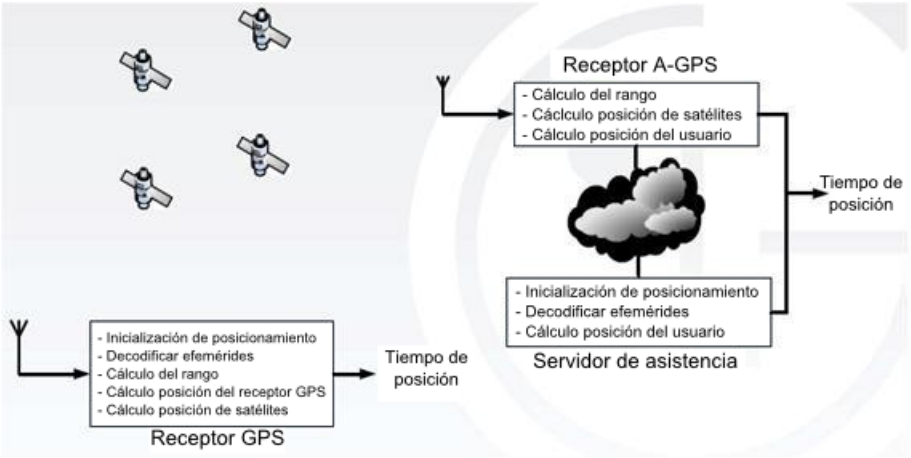


Figura 2.2: Arquitectura GPS frente a arquitectura A-GPS

2.1.2. Posicionamiento mediante la red de telefonía móvil

Las redes celulares se han extendido durante los últimos diez años y casi han alcanzado cobertura global. Actualmente, existen principalmente dos

redes celulares: sistema global para comunicaciones móviles (Global System for Mobile Communication, GSM) y sistema universal de telecomunicaciones móviles (Universal Telecommunication System, UMTS), también conocidas como 2G y 3G, mientras que la siguiente generación, 4G (Long Term Evolution, LTE), ya ha sido implantada en varios países.

A continuación en la figura 2.3 se muestra un esquema simplificado de la arquitectura GSM de acuerdo con [14].

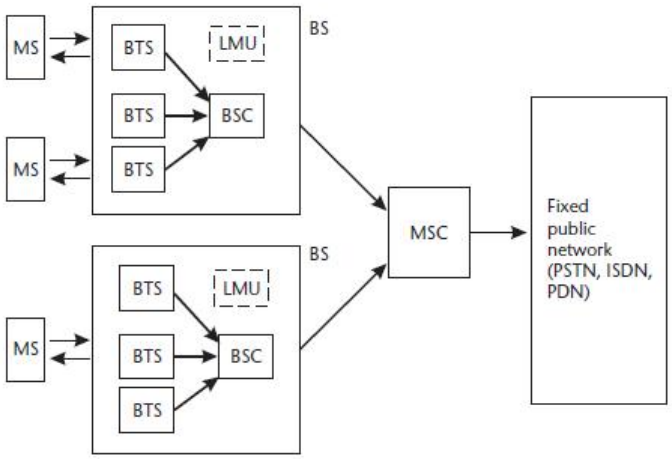


Figura 2.3: Arquitectura GSM, tomada de [14]

El terminal móvil (Mobile Station, MS) se comunica directamente con la estación base receptora (Base Transceiver Station, BTS). El subsistema de la estación base (Base Station, BS) incluye al receptor de la estación base y al controlador de la estación base (Base Station Controller, BSC) que se encarga de gobernar los parámetros de la interfaz aire de la comunicación en la celda o en un sector de la celda, incluyendo la frecuencia y el control de potencia. Aquellas redes que contengan el servicio de localización basado en la diferencia del tiempo de llegada (Time Difference of Arrival, TDOA) incluyen un subsistema llamado unidad de medida de la localización (Location Measurement Unit, LMU) que se encarga de recoger los tiempos de llegada de los datos enviados por el terminal móvil. El centro de conmutación móvil (Mobile Switch Center, MSC) funciona como una pasarela hacia la red de telefonía conmutada (Public Switched Telephone Network, PSTN), hacia la red digital de servicios integrados (Integrated Services Digital Network, ISDN) y la red de paquetes de datos (Public Data Network, PDN). Además, proporciona acceso a los registros de localización en los que se almacenan datos sobre la localización de los terminales móviles y se encargan

de la autenticación.

El intercambio de datos entre el terminal móvil y la estación base se realiza sobre canales físicos que se clasifican en canales de tráfico y canales de control. Los canales de tráfico contienen la información transmitida entre el terminal móvil y otro terminal móvil que se encuentre en la misma red o en otra, una vez se ha establecido la llamada. Los canales de control se mantienen entre las estaciones móviles y las estaciones base para establecer el final de la llamada.

De acuerdo con [16] existen tres arquitecturas que se pueden utilizar para posicionar teléfonos móviles: basadas en el terminal móvil (Mobile-based, MB), basadas en la red móvil (Network-based, NB) e híbridas. Debido a que uno de los objetivos de este proyecto es desarrollar una aplicación que se pueda instalar en dispositivos móviles y que permita conocer la posición del usuario mediante distintas técnicas de posicionamiento de forma autónoma, nos centraremos en la arquitectura conocida como MB. Esta arquitectura se caracteriza por la capacidad que tiene el terminal móvil de calcular su posición en función de las señales recibidas de las estaciones base. Esta técnica presenta varias ventajas según [14]:

- Cuando la información de posición es manejada por el sujeto en sí, esta técnica es la más segura. La información de localización y de seguimiento del sujeto no están disponibles en la red.
- Las capacidades de la red no están involucradas. Este sistema no hace uso de las facilidades y recursos de la red, y por tanto, su capacidad no se ve afectada.
- El número de medidas que se puedan tomar no está limitado por la red.

Existen múltiples métodos para calcular la posición del terminal móvil en función de la información proporcionada por la red, a continuación se nombran las técnicas más importantes en este ámbito hasta el momento.

Técnicas basadas en el indetificador de celda

En [21] se argumenta que las redes celulares se encuentran divididas en celdas que idealmente tienen una forma circular. Cada una de estas celdas se asocia a una estación base, de modo que, según el usuario se va desplazando a través de celdas, su teléfono móvil se conectará a la estación base de la que reciba mayor potencia de señal, que en muchos casos será la estación base más cercana al usuario.

La forma más básica de posicionamiento celular es utilizando la localización de la estación base a la que estamos conectados, cuya posición debe ser conocida de antemano y estar almacenada en una base de datos. De acuerdo con [17], este método se conoce como la celda de origen (Cell Of Origin, COO). La precisión de este método depende del tamaño de las celdas, que en el caso de las redes GSM, es de hasta 35 km de radio, mientras que en el caso de las redes 3G, la precisión es mayor puesto que las celdas son más pequeñas.

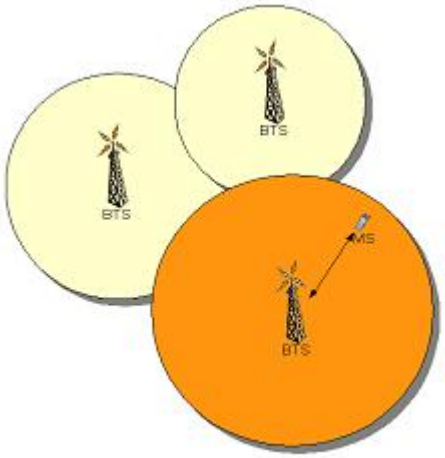


Figura 2.4: Esquema técnico COO, tomado de [17]

Este método puede mejorar su precisión disponiendo de la posición de varias estaciones base cercanas al móvil y, mediante algoritmos de trilateración que se explicarán más adelante, calculando la posición del móvil dentro de la celda en la que se sitúa.

Existe una mejora de este método, que viene dada por la utilización de varias antenas que permiten dividir dicho círculo en sectores, y por tanto, delimitar el área en la que el usuario se sitúa. Esta técnica se conoce como identificador de celda mejorado (Enhaced Cell-ID, E-CID) o celda de origen con sector de celda (COO with Cell Sector, COO CS) y su precisión también varía entre los 100m y los 35km, aunque el ángulo de cobertura dentro de la celda se reduce a 120º y por tanto, para poder calcular la posición del terminal móvil, debemos tener acceso a la información de los ángulos de apertura de las antenas de las estaciones base.

Una de las principales ventajas de esta técnica es que siempre que haya cobertura móvil se podrá utilizar, y el consumo de batería y el TTFF son menores que en el caso de GPS. Además debido a su reducida complejidad,

el coste de esta tecnología se ve fuertemente beneficiado.

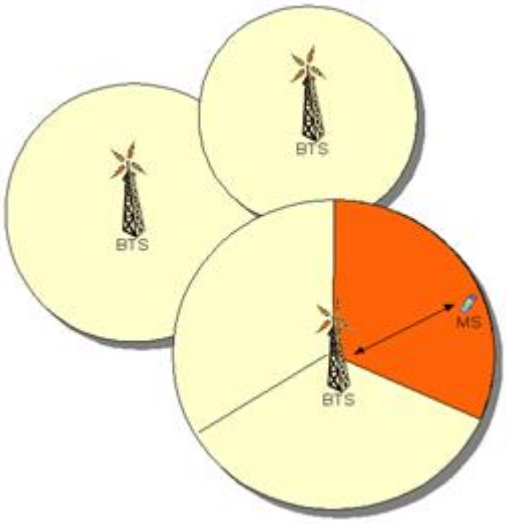


Figura 2.5: Esquema técnica COO CS, tomado de [17]

Técnica del avance de tiempo

Puesto que la precisión del método Cell-ID no es muy alta, dicha técnica se combina con la técnica de avance de tiempo (Cell-ID Timing Advance, TA). Esta técnica, según se describe en [21], mide el tiempo que tardan las señales en llegar desde el terminal móvil a la estación base, para que de este modo el terminal móvil pueda localizarse dentro de un círculo trazado alrededor de la estación base.

Según se comenta en [1], las redes GSM emplean acceso múltiple por división de tiempo (Time Division Multiple Access, TDMA) lo que permite que varios usuarios puedan compartir la misma frecuencia. Cada usuario tiene una ranura de tiempo asignada. El valor de TA es calculado por la red para la celda a la que el usuario está conectado y se basa en la distancia a la que el usuario está de la estación base de dicha celda. Este valor le indica al terminal móvil en qué instante puede transmitir dentro de la ranura de tiempo que se le ha reservado. Este dato en combinación con la posición geográfica de la estación base de la celda, nos permite obtener la posición del usuario. En la figura 2.6 se presenta un esquema del área de una celda dividida en secciones en función del valor que puede tomar el TA.

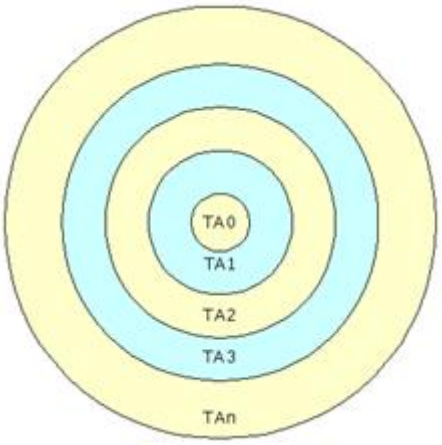


Figura 2.6: Esquema de la técnica TA, tomado de [17]

De acuerdo con [17] los valores están en el rango de 0 - 63, que equivale a 0m - 35 km., donde cada unidad son aproximadamente 3,69 microsegundos, equivalentes a 550m.

El resto de prestaciones (batería, latencia y disponibilidad) son las mismas que en el método Cell-ID, y se trata de una tecnología sólo disponible para GSM, y no aplicable a 3G. Además se necesita la participación por parte de la red celular para poder determinar la posición del usuario y no sólo esto afecta de forma negativa a su complejidad, sino también el hecho de que los tiempos, al ser muy pequeños, no se adecuan a la precisión de cálculo que puede soportar el sistema operativo un dispositivo móvil.

Técnica basada en la diferencia del tiempo de llegada

Esta técnica (Time Difference of Arrival, TDOA) se basa en estimar la diferencia del tiempo que tarda en llegar la señal de la estación base al terminal móvil. Normalmente, según se expone en [11], esta estimación se lleva a cabo realizando una captura de la señal recibida en varios receptores en el mismo instante de tiempo.

Para una diferencia de tiempo concreta, se define una hipérbola entre dos receptores donde la estación móvil puede estar, asumiendo que la estación base y el móvil se encuentran en el mismo plano. Este proceso se repite con otro receptor combinándolo con los dos anteriores, y se traza una nueva hipérbola, tal y como se representa en la figura 2.7. El punto en el que intersectan las dos hipérbolas se considera la posición del terminal móvil.

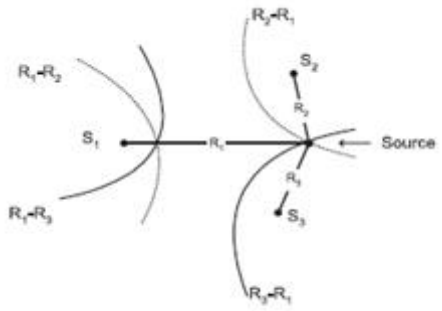


Figura 2.7: Esquema de las hipérbolas trazadas, tomado de [17]

Figura 2.8: Esquema de la técnica TDOA, tomado de [17]

Existe una mejora de este método que se basa en que dos o más estaciones base envían señales de forma simultánea al terminal móvil y éste calcula la diferencia de tiempo a cada estación base, y lo traduce a términos de distancia entre él mismo y cada una de las estaciones base. A este método se le denomina mejora de la diferencia de tiempo observada (Enhanced Observed Time Difference, EOTD).

Figura 2.9: Esquema de la técnica EOTD, tomado de [17]

En resumen, la estación base debe estar correctamente sincronizada con el terminal móvil y un retardo de un segundo implica un error de precisión de 150m, lo que aumenta de forma exponencial la complejidad de estos modelos aunque la precisión sea mejor que en las técnicas anteriores, mientras que el consumo de batería, la latencia, la disponibilidad y el coste de este servicio mantienen sus prestaciones frente al resto de modelos.

Sin embargo, ninguna de las dos versiones de este método puede aplicarse dentro del ámbito de nuestro proyecto puesto que sólo utilizaremos un terminal móvil receptor y las estaciones base pertenecen a terceros, y por tanto, no podremos ejercer ningún control sobre ellas para conseguir que transmitan una señal en el mismo instante.

Técnica del ángulo de llegada

En esta técnica (Angle of Arrival, AOA), en lugar de utilizar distancias, se emplean ángulos para determinar la posición del términal móvil. Para calcular una posición en dos dimensiones necesitamos dos ángulos y una distancia, como puede ser la distancia que separa dos estaciones. También se debe definir un ángulo de referencia, normalmente, se hacen coincidir los 0º con el norte magnético, tal y como se representa en la figura 2.10.

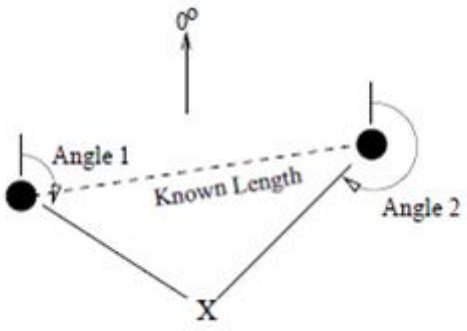


Figura 2.10: Esquema de los ángulos definidos, tomado de [17]

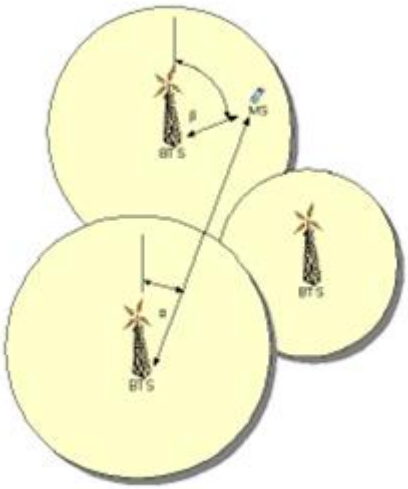


Figura 2.11: Esquema de la técnica AOA, tomado de [17]

La precisión de este método se sitúa dentro de un rango de 50m - 150m, pero se necesita una antena que nos porporcione información de sus ángulos de apertura como unidad de medida, según se menciona en [17], lo cual afecta negativamente al coste incrementándolo. Sin embargo, el consumo de batería, la latencia y la disponibilidad de este servicio mantienen sus prestaciones frente al resto de modelos.

Por lo tanto, al necesitar una antena especial que nos proporcione dichas medidas, esta técnica también queda fuera del ámbito de este proyecto.

Técnica del tiempo de vuelo

Esta técnica (Round-Trip-Time-of-Flight, RToF) consiste en medir el tiempo de propagación de la señal enviada del terminal móvil a la estación

base. La estación base envía una señal al móvil y espera a recibir la respuesta por parte del móvil, tal y como se muestra en la figura 2.12.

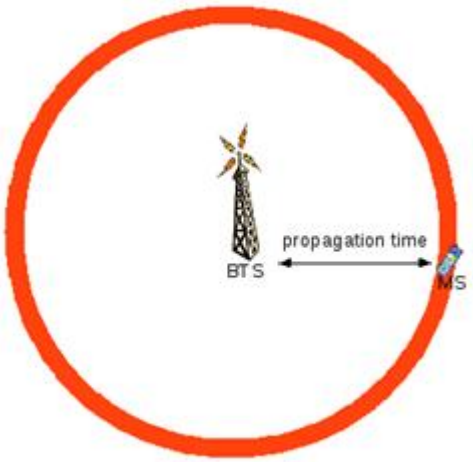


Figura 2.12: Esquema de la comunicación entre la estación base y el terminal móvil, tomado de [17]

Figura 2.13: Esquema de la técnica RToF, tomado de [17]

Se necesita al menos la señal de tres estaciones base, y la precisión depende del retardo en la comunicación entre la estación base y el móvil, según se comenta en [17], normalmente se encuentra dentro de un rango de 50m a 150m. El resto de prestaciones (coste, disponibilidad, latencia y consumo de batería) se mantienen con respecto al resto de modelos, sin embargo, la complejidad aumenta puesto que el tiempo de propagación se encuentra expresado en unidades muy pequeñas y esto afecta a la carga computacional.

Técnica basada en el indicador de potencia de señal recibida

Esta técnica (Received Signal Strength Indicator, RSSI), también se conoce como con el nombre de COO CS and RSS, y se trata de medir el nivel de potencia de la señal recibida por el terminal móvil dentro de la celda en la que se localiza, aumentando así la precisión a un rango de 100m a 20 km.

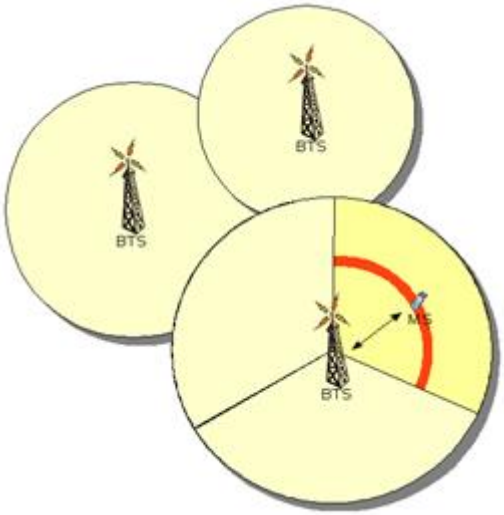


Figura 2.14: Esquema de la técnica RSSI, tomado de [17]

Uno de los principales inconvenientes de esta técnica es que la potencia recibida se ve afectada por otros factores como son la reflexión y el efecto multitrayecto, y la absorción de las paredes de los edificios, lo que dificulta su utilización en entornos cerrados.

Resumen de técnicas de posicionamiento basadas en la red celular

Resumiendo, la precisión de los métodos de posicionamiento mediante celdas es menor que la obtenida mediante un posicionamiento GPS, sin embargo, el consumo de batería es menor. Además, estos métodos también permiten el posicionamiento en interiores y alguno de ellos no necesita instalar un nuevo hardware en el dispositivo móvil para poder utilizarlas, lo que influye positivamente en el coste. En la siguiente tabla 2.1, se recoge la precisión de cada uno de los métodos vistos:

Método	Precisión
Cell Of Origin	100m - 35 km
COO with Cell Sector	100m - 35km
COO CS and Received Signal Strength	100m - 20 km
Angle Of Arrival	50m - 150m
Round-Trip-Time-of-Flight	50 - 150m
Time Difference Of Arrival	50 - 150m
Enhanced Observed Time Difference	50m - 150m

Tabla 2.1: Precisión de las disitntas tecnologías de posicionamiento mediante celdas

2.1.3. Posicionamiento mediante estaciones WiFi

El posicionamiento mediante estaciones WiFi es muy similar al posicionamiento mediante CID, pero en lugar de conocer la localización de estaciones base, se conoce la localización de puntos de acceso WiFi (Access Point, AP). Como factor de identificación único, se utiliza la dirección MAC de estos puntos de acceso.

Las redes WiFi se caracterizan por su corto alcance (60m aproximadamente según se comenta en [21]) comparado con las redes anteriormente mencionadas, lo que se traduce en una mayor precisión. Además el TTFF es bastante menor puesto que no se necesita el establecimiento de conexión a redes WiFi y la búsqueda en la base de datos apenas conlleva pocos segundos.

Estas técnicas de posicionamiento en un principio sólo se consideraron para interiores, pero el hecho de que cada vez más puntos de acceso WiFi están siendo instalados en las ciudades, hace que estas técnicas sean cada vez más importantes. Por otra parte, el consumo de batería queda restringido a la carga de procesamiento derivada del cómputo de la posición del usuario.

De acuerdo con [21], las técnicas de posicionamiento mediante WiFi se pueden dividir principalmente en activas o en pasivas. Las activas son aquellas en que el cálculo de la posición del términal móvil lo delegan en los puntos de acceso, mientras que las técnicas pasivas el cálculo de la posición lo realiza el terminal móvil. En nuestro caso nos centraremos en estas últimas.

Las técnicas de posicionamiento mediante comunicaciones WiFi se pueden dividir en las siguientes tres categorías.

Sensores de proximidad

En este caso, la posición del punto de acceso más cercano es la que se adopta como localización del usuario. De acuerdo con [20], dentro de los métodos existentes el que se encuentra dentro del marco de nuestro proyecto es la monitorización de puntos de acceso. Este método se utiliza cuando el dispositivo móvil se encuentra en un rango alcanzable entre distintos puntos de acceso. En la figura, los objetos X, Y y Z se localizan monitorizando su conectividad a uno o varios puntos de acceso en una red inalámbrica, y adoptan como posición la posición del punto de acceso del que reciben mayor potencia. El radio de cobertura de las estaciones depende de la tecnología inalámbrica empleada, por ejemplo, si se empleasen rayos infrarrojos, la forma de la celda tendría la geometría de la habitación en la que nos situemos.

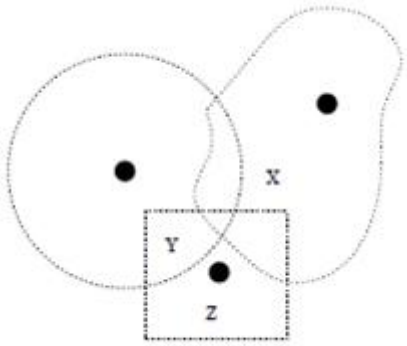


Figura 2.15: Esquema tecnología Proximity Sensing, tomado de [20]

Esta técnica presenta la ventaja de que es la más sencilla, sin embargo, la precisión puede variar en función de la cobertura de los puntos de acceso que puede llegar a variar entre 60m y 100m. En cuanto al coste es mínimo, puesto que sólo se necesita tener una tarjeta WiFi en el móvil que permita dicha conexión, algo que la mayoría de los teléfonos actuales tienen.

Lateración

Al igual que en el posicionamiento celular, en las comunicaciones WiFi se pueden aplicar cálculos de lateración para determinar la posición del usuario conociendo la situación de los puntos de acceso WiFi. Por lo tanto, las técnicas de lateración que se presentarán en la siguiente sección también son aplicables aquí.

Fingerprinting

Esta técnica [20] hace uso de una base de datos donde se almacenan patrones de nivel de potencia recibida en diferentes puntos del lugar a caracterizar, obtenidos en una fase previa de medición, y el nivel de potencia actual recibida de los puntos de acceso. Por lo tanto, el patrón (la posición donde se tomó la medida) que más se parezca a la medida actual será considerado como la posición del usuario.

En este caso, se necesita tener un mapa de la zona en la cual el nivel de potencia de la señal para una determinada posición esté almacenado. Esta información puede estar almacenada durante una fase previa en el móvil, o bien en la red, reduciendo así el consumo de memoria en el dispositivo móvil. En la fase de evaluación, el usuario mide el nivel de potencia recibida y el punto medido con anterioridad que se encuentre más cerca del usuario es el elegido como el punto de posicionamiento del usuario.

Este modelo tampoco se adapta a nuestras necesidades puesto que la aplicación a desarrollar tiene como objetivo determinar nuestra posición en tiempo real, sin previamente realizar una fase de entrenamiento de nuestro modelo.

2.2. Métodos de lateración

Como se ha mencionado antes, las técnicas de posicionamiento mediante celdas pueden mejorar su precisión combinándolas con métodos de lateración. En dichos métodos, la distancia entre uno o varios puntos de acceso y el terminal móvil es lo que se determina. Para calcular la posición de un objeto en dos dimensiones se requiere el cálculo de la distancia a 3 puntos no colindantes.

De acuerdo con [20], existen cuatro métodos principales en la lateración:

- Directa. Son medidas tomadas físicamente, como puede ser usando una cinta métrica. Esta técnica queda fuera del ámbito de nuestro proyecto por motivos obvios.

- Tiempo de vuelo (TOF). Este método es muy parecido a la técnica nombrada en el apartado anterior, TOA, puesto que se basa en medir el tiempo que tarda en llegar una señal de un punto a otro con una velocidad conocida. El principal inconveniente de este método es que se necesitan relojes de alta resolución y los relojes del terminal móvil y el punto de acceso deben estar sincronizados. Sólo en el caso de que se mida el tiempo de ida y vuelta (Round Trip Time, RTT) en el receptor, su reloj deberá mantener la precisión durante la medida.

- Atenuación. Sabemos que la intensidad de la señal emitida decrece con la distancia. Conociendo la función que relaciona la atenuación y la distancia y la potencia de emisión, es posible estimar la distancia de un objeto a un punto midiendo el nivel de potencia recibida. Por ejemplo, en espacio libre la señal es atenuada en un factor proporcional a r^{-2}, donde r es la distancia entre el móvil y el punto de acceso. Sin embargo, en un entorno cerrado este método es menos preciso debido a la presencia de obstáculos. Las reflexiones, la refracción y el multitrayecto convierten a este método en impreciso a la hora de calcular la distancia.

- Métodos de Trilateración. Conociendo las coordenadas de varios puntos de acceso podemos calcular la posición del usuario empleando algoritmos de triangulación, que veremos más adelante.

2.2.1. Métodos de trilateración

Anterioremente se comentaba que las técnicas de posicionamiento podrían mejorar su precisión haciendo uso de métodos de trilateración. Dichos métodos pueden ser aplicados siempre que sean conocidas tres posiciones cardinales de tres estaciones base cercanas al usuario, que normalmente serán: la estación base a la que el usuario está conectado, y las dos estaciones base vecinas con el nivel de potencia recibida más alto.

Estos métodos de trilateración tienen como objetivo calcular el centro de un triángulo, y en nuestro caso consideraremos dicho punto como la posición del usuario. Existen bastantes tipos de centros de un triángulo que pueden ser consultados en la enciclopedia de los centros de los triángulos [22], sin embargo, nos centraremos en los cuatro centros más populares del triángulo.

A diferencia de los paralelogramas y los círculos, los triángulos tienen distintos tipos de centros, cuatro de los cuales fueron definidos por los griegos siglos atrás, y se conocen como los cuatro centros clásicos: el centroide, el ortocentro, el incentro y el circuncentro, de acuerdo con [12].

El centroide, el ortocentro y el circuncentro son colineares, es decir, se sitúan sobre la misma recta que recibe el nombre de recta de Euler.

- El centroide de un triángulo es el punto de concurrencia de las medianas de un triángulo. Este punto también es conocido como el baricentro del triángulo.

- El ortocentro de un triángulo es el punto de concurrencia de las alturas del triángulo.

- El incentro del triángulo es el punto de concurrencia de los bisectores de los ángulos del triángulo. Equivale al punto en el interior del triángulo que es equidistante de los tres lados del triángulo.

- El circuncentro del triángulo es el punto en el plano del triángulo equidistante a los tres vértices del triángulo.

A la hora de calcular nuestra posición como centro de un triángulo formado por tres estaciones base, vamos a utilizar lo que se conoce como coordenadas baricéntricas.

El problema que presentan las coordenadas cartesianas es que las coordenadas de cada uno de los vértices del triángulo deben estar referenciadas a un origen de coordenadas, y situadas dentro de dos ejes perpendiculares que parten de él. Esto es un inconveniente puesto que las coordenadas que nosotros utilizaremos serán la latitud y la longitud correspondientes a la situación de cada una de las estaciones base, y lo que nos interesa es calcular nuestra posición en relación a estas tres estaciones tomando como origen el centro de gravedad de la Tierra. Este problema queda solventando con el uso de coordenadas baricéntricas.

De acuerdo con [12], la palabra baricentro significa centro de gravedad. En 1827 Möbius introdujo con su libro la deficinión del sistema de coordenadas baricéntricas. Su idea era que para un triángulo dado, se asignan unas masas m_1, m_2, m_3 a tres puntos no colineares A_1, A_2, A_3, y se considera el centro de masa, llamado baricentro, según la ecuación:

$$P = \frac{m_1 A_1 + m_2 A_2 + m_3 A_3}{m_1 + m_2 + m_3} \tag{2.1}$$

Estas masas, o también llamados pesos, tendrán distintos valores para cada uno de los cuatro centros del triángulo nombrados anteriormente.

Centroide

De acuerdo con lo expuesto anteriormente en la ecuación 2.1, la fórmula de las coordenadas del centroide seguirá la siguiente expresión:

$$G = \frac{m_1 A_1 + m_2 A_2 + m_3 A_3}{m_1 + m_2 + m_3} \tag{2.2}$$

Como el centroide es el punto donde las medianas de los lados del triángulo se juntan, partimos del cálculo de los puntos medios de cada uno de los lados como se muestra en la siguiente figura 2.16

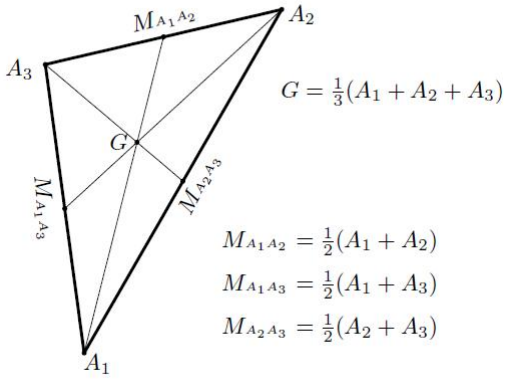


Figura 2.16: Puntos medios de los lados y el centroide G de un triángulo, tomada de [12]

Las ecuaciones obtenidas mediante permutaciones cíclicas y que definen cada una de las medianas de los lados del triángulo en función de dichos índices de permutación son las siguientes:

$$\begin{aligned} L_{123}(t_1) &= \tfrac{t_1}{2}A_1 + \tfrac{t_1}{2}A_2 + (1-t_1)A_3 \\ L_{231}(t_2) &= \tfrac{t_2}{2}A_2 + \tfrac{t_2}{2}A_3 + (1-t_2)A_1 \\ L_{312}(t_3) &= \tfrac{t_3}{2}A_3 + \tfrac{t_3}{2}A_1 + (1-t_3)A_2 \end{aligned} \tag{2.3}$$

Para obtener el punto donde se cortan, simplemente hay que resolver la ecuación $L_{312}(t_3) = L_{123}(t_1) = L_{231}(t_2)$ para las incógnitas t_3, t_2, t_1, obteniendo $t_3 = t_2 = t_1 = 2/3$.

Por tanto el centroide será:

$$G = \frac{A_1 + A_2 + A_3}{3} \tag{2.4}$$

Comparándola con nuestra ecuación inicial 2.2 e identificando términos:

$$m_3 = m_2 = m_1 = \frac{1}{3} \tag{2.5}$$

Cabe resaltar que el centroide siempre se sitúa dentro del triángulo sin importar la forma que éste tenga.

Ortocentro

El ortocentro, tal y como se muestra en la figura 2.17, es el punto en el que se intersectan las alturas de cada uno de los lados de un triángulo, y sus coordenadas vendrán dadas por la ecuación 2.6:

$$H = \frac{m_1 A_1 + m_2 A_2 + m_3 A_3}{m_1 + m_2 + m_3} \tag{2.6}$$

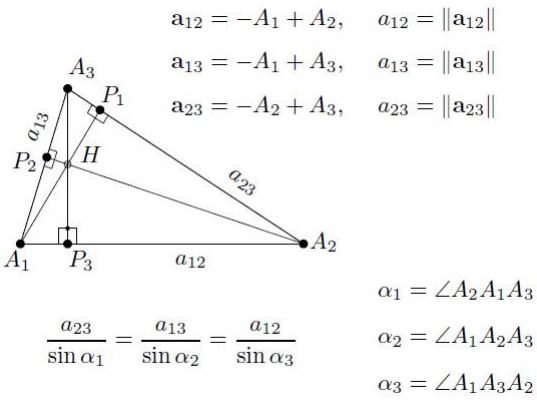


Figura 2.17: El ortocentro de un triángulo, tomada de [12]

Los puntos P_1, P_2, P_3 son los puntos de corte de las alturas, y sus coordenadas baricéntricas vienen dadas por las siguientes expresiones:

$$\begin{aligned} P_1 &= \tfrac{1}{2}\left(\tfrac{-a_{12}^2+a_{13}^2+a_{23}^2}{a_{23}^2}\right) A_2 + \tfrac{1}{2}\left(\tfrac{a_{12}^2-a_{13}^2+a_{23}^2}{a_{23}^2}\right) A_3 \\ P_2 &= \tfrac{1}{2}\left(\tfrac{-a_{12}^2+a_{13}^2+a_{23}^2}{a_{13}^2}\right) A_1 + \tfrac{1}{2}\left(\tfrac{a_{12}^2+a_{13}^2-a_{23}^2}{a_{13}^2}\right) A_3 \\ P_3 &= \tfrac{1}{2}\left(\tfrac{a_{12}^2-a_{13}^2+a_{23}^2}{a_{12}^2}\right) A_1 + \tfrac{1}{2}\left(\tfrac{a_{12}^2+a_{13}^2-a_{23}^2}{a_{12}^2}\right) A_2 \end{aligned} \tag{2.7}$$

Luego, las ecuaciones que definen las alturas de cada lado del triángulo son las siguientes:

$$\begin{aligned} L_{A_1P_1} &= A_1 + (-A_1 + P_1)t_1 \\ L_{A_2P_2} &= A_2 + (-A_2 + P_2)t_2 \\ L_{A_3P_3} &= A_3 + (-A_3 + P_3)t_3 \end{aligned} \tag{2.8}$$

Para encontrar el punto de corte de las tres altitudes, debemos resolver la siguiente ecuación:

$$L_{A_1P_1} = A_1+(-A_1+P_1)t_1 = L_{A_2P_2} = A_2+(-A_2+P_2)t_2 = L_{A_3P_3} = A_3+(-A_3+P_3)t_3 \tag{2.9}$$

Obteniendo:

$$\begin{aligned} t_1 &= \tfrac{2(-a_{12}+a_{13}+a_{23})}{D} a_{23} \\ t_2 &= \tfrac{2(-a_{12}+a_{13}-a_{23})}{D} a_{13} \\ t_3 &= \tfrac{2(a_{12}-a_{13}-a_{23})}{D} a_{12} \end{aligned} \tag{2.10}$$

donde

$$D = a_{12}^2 + a_{13}^2 + a_{23}^2 - 2(a_{12}a_{13} + a_{12}a_{23} + a_{13}a_{23}) \tag{2.11}$$

Sustituyendo t_1, t_2, t_3 en la ecuación 2.8 e identificando términos con la ecuación 2.6, la coordenadas baricéntricas que se obtienen son:

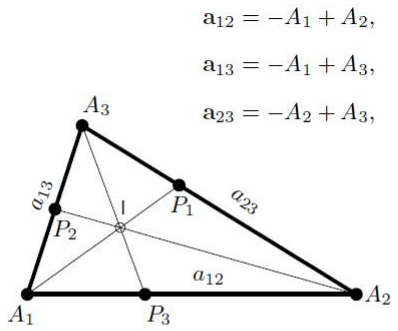
$$\begin{aligned} m_1 &= a_{23}^2 - (a_{12}^2 - a_{13}^2)^2 \\ m_2 &= a_{13}^2 - (a_{12}^2 - a_{23}^2)^2 \\ m_3 &= a_{12}^2 - (a_{23}^2 - a_{13}^2)^2 \end{aligned} \tag{2.12}$$

A diferencia del centroide, el ortocentro no se encuentra siempre en el interior del triángulo, puede situarse fuera como en el caso de un triángulo obtuso o incluso puede caer en uno de los lados del triángulo como en el caso del triángulo rectángulo.

Incentro

El incentro de un triángulo es el centro del cículo interior y tangente a cada uno de sus lados y vendrá dado por las coordenadas baricéntricas de la ecuación 2.13.

$$I = \frac{m_1 A_1 + m_2 A_2 + m_3 A_3}{m_1 + m_2 + m_3} \tag{2.13}$$



$$\mathbf{a}_{12} = -A_1 + A_2, \qquad a_{12} = \|\mathbf{a}_{12}\|$$
$$\mathbf{a}_{13} = -A_1 + A_3, \qquad a_{13} = \|\mathbf{a}_{13}\|$$
$$\mathbf{a}_{23} = -A_2 + A_3, \qquad a_{23} = \|\mathbf{a}_{23}\|$$

$$\angle A_2A_1P_1 = \angle A_3A_1P_1$$
$$\angle A_1A_2P_2 = \angle A_3A_2P_2$$
$$\angle A_1A_3P_3 = \angle A_2A_3P_3$$

$$\mathbf{p}_1 = -A_1 + P_1, \qquad p_1 = \|\mathbf{p}_1\|$$
$$\mathbf{p}_2 = -A_2 + P_2, \qquad p_2 = \|\mathbf{p}_2\|$$
$$\mathbf{p}_3 = -A_3 + P_3, \qquad p_3 = \|\mathbf{p}_3\|$$

$$\alpha_1 = \angle A_2A_1A_3, \quad \alpha_2 = \angle A_1A_2A_3, \quad \alpha_3 = \angle A_1A_3A_2$$

Figura 2.18: Incentro de un triángulo, tomada de [12]

Tal y como se muestra en la figura 2.18 Los puntos P_1, P_2, P_3 son los puntos de corte de los bisectores, y sus coordenadas baricéntricas vienen dadas por las siguientes expresiones:

$$\begin{array}{l} P_1 = \frac{a_{13}A_2+a_{12}A_3}{a_{13}+a_{12}} \\ P_2 = \frac{a_{23}A_1+a_{12}A_3}{a_{23}+a_{12}} \\ P_3 = \frac{a_{23}A_1+a_{13}A_2}{a_{23}+a_{13}} \end{array} \tag{2.14}$$

Luego, las ecuaciones que definen los bisectores de cada ángulo del triángulo son las siguientes:

$$\begin{array}{l} L_{A_1P_1} = A_1 + (-A_1 + P_1)t_1 \\ L_{A_2P_2} = A_2 + (-A_2 + P_2)t_2 \\ L_{A_3P_3} = A_3 + (-A_3 + P_3)t_3 \end{array} \tag{2.15}$$

Para encontrar el punto de corte de los tres bisiectores, debemos resolver la siguiente ecuación:

$$L_{A_1P_1} = A_1+(-A_1+P_1)t_1 = L_{A_2P_2} = A_2+(-A_2+P_2)t_2 = L_{A_3P_3} = A_3+(-A_3+P_3)t_3 \tag{2.16}$$

obteniendo:

$$\begin{array}{l} t_1 = \frac{a_{12}+a_{13}}{a_{12}+a_{13}+a_{23}} \\ t_2 = \frac{a_{12}+a_{23}}{a_{12}+a_{13}+a_{23}} \\ t_3 = \frac{a_{13}+a_{23}}{a_{12}+a_{13}+a_{23}} \end{array} \tag{2.17}$$

Sustituyendo t_1, t_2, t_3 en la ecuación 2.15 e identificando términos con la ecuación 2.13, la coordenadas baricéntricas que se obtienen son:

$$\begin{array}{l} m_1 = a_{23} \\ m_2 = a_{13} \\ m_3 = a_{12} \end{array} \tag{2.18}$$

Al igual que el centroide, el incentro siempre se sitúa dentro de los límites del triángulo sin importar su forma.

Circuncentro

El circuncentro de un triángulo es el centro del círculo que pasa por los tres vértices que componen dicho triángulo, y sus coordenadas siguen la ecuación:

$$O = \frac{m_1A_1 + m_2A_2 + m_3A_3}{m_1 + m_2 + m_3} \tag{2.19}$$

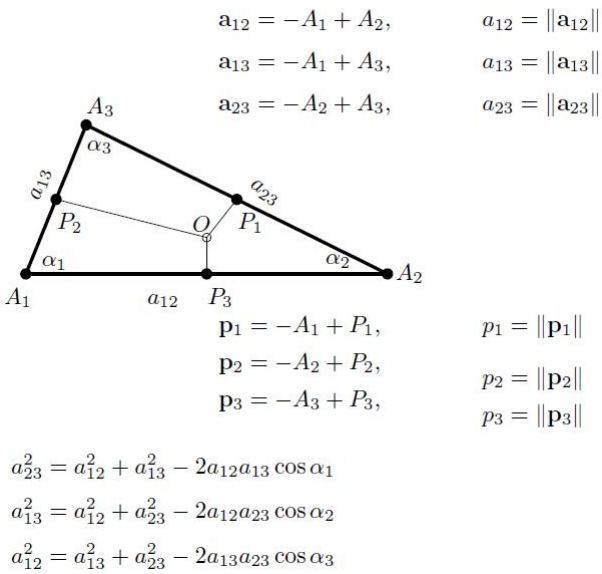


Figura 2.19: Incentro de un triángulo, tomada de [12]

Como se puede observar en la figura 2.19, el circuncentro puede ser calculado como el punto de corte entre los bisectores perpendiculares, por lo tanto, debe cumplirse la siguiente condición:

$$\begin{aligned} \|-A_1 + O\|^2 &= \|-A_2 + O\|^2 \\ \|-A_2 + O\|^2 &= \|-A_3 + O\|^2 \end{aligned} \tag{2.20}$$

Y con la norma de normalización de las coordenadas baricéntricas:

$$m_1 + m_2 + m_3 = 1 \tag{2.21}$$

Sustituyendo la ecuación de normalización en las ecuaciones 2.20 y simplificando, obtenemos una ecuación que depende linealmente de m_1 y m_2.

$$a_{13}^2 - a_{23}^2 - m_1(a_{12}^2 + a_{13}^2 - a_{23}^2) + m_2(a_{12}^2 - a_{13}^2 + a_{23}^2) = 0 \tag{2.22}$$

Al aplicar permutación cíclica sobre los vértices obtenemos un sistema de ecuaciones que depende linealmente de las tres incógnitas m_1,m_2 y m_3.

$$\begin{aligned} &a_{13}^2 - a_{23}^2 - m_1(a_{12}^2 + a_{13}^2 - a_{23}^2) + m_2(a_{12}^2 - a_{13}^2 + a_{23}^2) = 0 \\ &a_{12}^2 - a_{13}^2 - m_2(a_{12}^2 - a_{13}^2 + a_{23}^2) + m_3(-a_{12}^2 + a_{13}^2 + a_{23}^2) = 0 \\ &m_1 + m_2 + m_3 = 0 \end{aligned} \tag{2.23}$$

Finalmente, mediante la resolución del sistema de coordenadas, obtenemos las coordenadas baricéntricas:

$$\begin{aligned} m_1 &= a_{23}^2(a_{12}^2 + a_{13}^2 - a_{23}^2) \\ m_2 &= a_{13}^2(a_{12}^2 - a_{13}^2 + a_{23}^2) \\ m_3 &= a_{12}^2(-a_{12}^2 + a_{13}^2 + a_{23}^2) \end{aligned} \tag{2.24}$$

Al igual que ocurría con el ortocentro, el circuncentro también puede situarse fuera del triángulo o sobre uno de los lados.

Relaciones entre los distintos centros

Existen una serie de relaciones entre los distintos centros, según se comenta en[18], que cabe resaltar:

- El centroide, el otrocentro y el circuncentro se encuentran en la misma línea.
- El centroide siempre se sitúa entre el ortocentro y el circuncentro.
- La distancia entre el centroide y el otrocentro siempre es dos veces la distancia entre el centroide y el circuncentro.
- En los triángulos obtusos, el circuncentro siempre se sitúa fuera opuesto al mayor ángulo del triángulo. El ortocentro siempre se situa fuera opuesto al lado más largo del triángulo, en el mismo lado que el ángulo más grande.
- La única vez que tres de estos centros caen en el mismo sitio es en el caso de un triángulo equilátero.

2.3. Plataforma Android

El objetivo de este proyecto es analizar las prestaciones de los métodos de trilateración que se emplean para calcular la posición del usuario y para ello se debe implementar una aplicación que nos proporcione dicha posición y que sea instalable en un dispositivo móvil. El sistema operativo para el cual se va a desarrollar la aplicación y que se ha escogido debido a su abundante presencia dentro del mercado de telefonía móvil, es el sistema operativo Android. Esta plataforma de desarrollo nos proporciona información relativa a la localización del dispositivo móvil, por ello, este capítulo tiene por objetivo presentar una breve descripción de las características más importantes de

este sistema operativo y de las herramientas que nos facilita para llevar a cabo nuestro objetivo.

Android es un sistema operativo además de ser una plataforma de *software* basada en el núcleo de Linux de acuerdo con [1]. Fue diseñada en un principio para dispositivos móviles, y permite controlar dispositivos por medio de bibliotecas desarrolladas, o incluso adaptadas, por Google mediante el lenguaje de programación Java.

Una de las principales ventajas de Android es que es una plataforma de código abierto, es decir, que cualquier desarrollador puede crear y desarrollar aplicaciones escritas con lenguaje C u otros lenguajes y compilarlas a código nativo ARM (API de Android).

Android es un sistema operativo que ya se estaba desarrollando en 2005, año en el que Google da un primer paso y compra este novedoso sistema operativo para móviles a sus desarrolladores. En 2007 se lanza el *Android Software Development Kit* (Android SDK), y prácticamente un año después aparecería una versión beta del SDK, la versión 0.9. Un mes más tarde se lanzó la versión Android 1.0 y seis meses después fue presentada la versión 1.1 con algunas modificaciones estéticas y nuevas posibilidades, como la búsqueda por voz, nuevas aplicaciones en el Android Market, etc.

En Mayo de 2009, Google lanza la versión 1.5 conocida como *Cupcake* que incluye grabación de video, soporte para Stereo Bluetooth, sistema de teclado personalizable en pantalla o reconocimiento de voz. En septiembre del mismo año surge Android 1.6, la versión llamada *Donut*, optimizando las búsquedas, añadiendo el indicador de uso de la batería y otras mejoras.

Más adelante, aparece Android 2.0 incluyendo aplicaciones precargadas que requerían un *hardware* más potente que la generación de móviles anteriores. Más tarde llega Android 2.1 conocida como *Éclair* que incluye capacidades como el 3D y fondos de pantalla en vivo.

A esta versión le han seguido Android 2.2 (*Froyo*) y Android 2.3 (*Gingerbread*). Pero ha sido con la aparición de las tabletas donde Google ha mejorado las prestaciones, sobre todo visuales, con sus versiones Android 3.0, Android 3.1, Android 3.2 y, muy recientemente, Android 4.0, conocida como *Ice scream*.

2.3.1. Arquitectura de Android

La arquitectura del sistema operativo Android se divide en los siguientes niveles.

- Aplicaciones. Entre las las aplicaciones creadas con la plataforma Android se incluyen como base un cliente de correo electrónico, un calen-

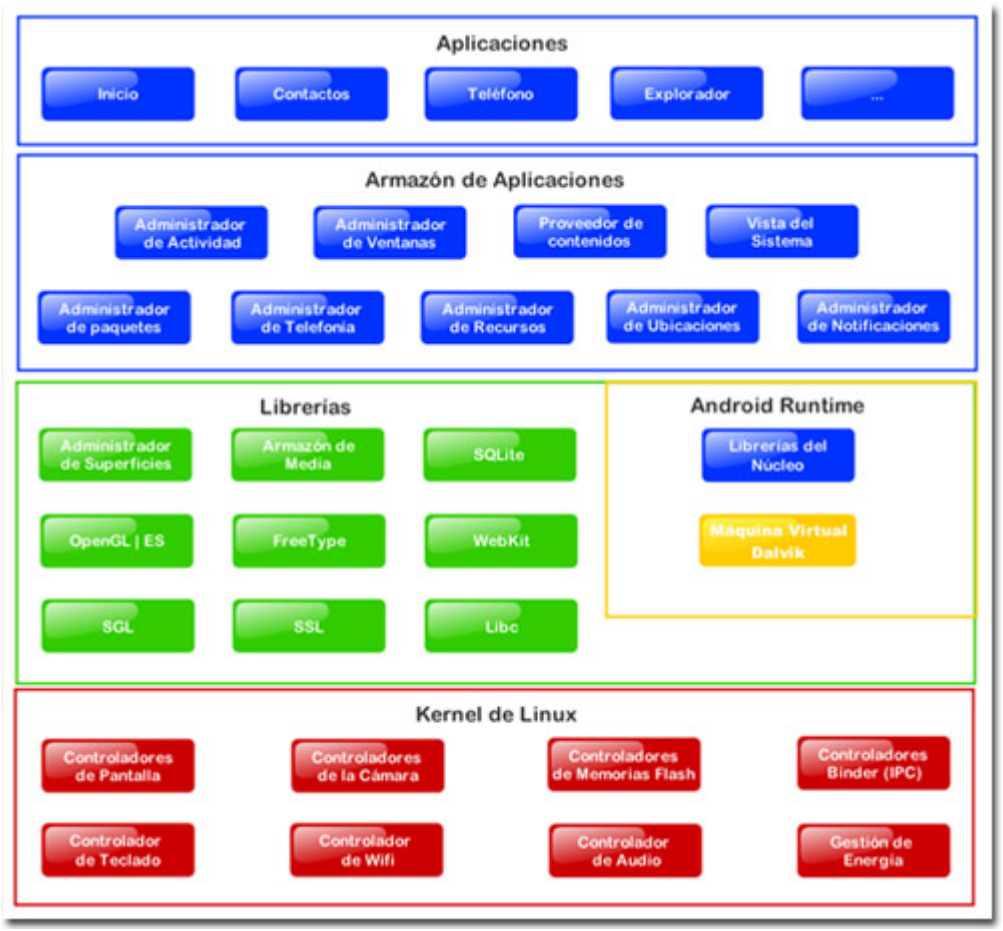


Figura 2.20: Arquitectura de Android, tomada de [1]

dario, un programa de mensajes de texto, mapas, un navegador web, contactos, y algunos otros servicios mínimos. En este nivel es donde situamos las aplicaciones de terceros, entre las cuales, se posiciona nuestra aplicación.

- El armazón de aplicaciones. Los desarrolladores de Android tienen acceso al código fuente usado en las aplicaciones base para evitar que se generen varias aplicaciones distintas que responden a la misma funcionalidad, y permitiendo así, que los programas sean modificados o reemplazados por cualquier usuario sin tener que empezar a programar sus aplicaciones desde cero. Algunas de estas API serán explicadas en mayor detalle más adelante.

- Librerías. Android incluye en su arquitectura un conjunto de librerías

C y C++, que son accesibles a los desarrolladores a través del armazón de aplicaciones. Dentro de las librerías de Android se pueden destacar las siguientes:

  - Libc: librería que incluye todas las cabeceras y funciones de acuerdo con el estándar del lenguaje C.
  - Administrador de Superficies: se encarga de gestionar las ventanas pertenecientes a las diferentes aplicaciones activas en cada momento, y de sus elementos de navegación.
  - OpenGL ES: es la librería que proporciona funciones para mejorar la capacidad gráfica de Android. Da sorpote a gráficos 2D y 3D siempre que el *hardware* del dispositivo lo permita.
  - SGL: proporciona gráficos 2D, y es por tanto, la más utilizada en Android.
  - Armazón multimedia: librería que contiene todos los estándares de codificación necesarios para grabar y reproducir el contenido multimedia en Android.
  - SSL: esta librería permite establecer conexiones mediante el protocolo de capa de protección segura (Secure Sockets Layer, SSL).
  - SQLite: permite la creación y la gestión de bases de datos dentro de un entorno Android.
  - WebKit: proporciona un motor para las aplicaciones de Android de tipo navegador.

- Entorno de ejecución de Android. Android incorpora un conjunto de librerías que aportan la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación Java. La máquina virtual está basada en registros, y ejecuta clases compiladas por el compilador de Java que anteriormente han sido transformadas al formato `.dex` (Dalvik Executable) por la herramienta *dx*.

- Kernel de Linux. Android utiliza la verisón 2.6 de Linux haciendo uso de servicios tales como seguridad, pila de red y controladores. Este kernel también actúa como un nivel intermedio en el *hardware* y el *software* del dispositivo.

2.3.2. Máquina Virtual Dalvik

De acuerdo con [1], Dalvik es la máquina virtual que utiliza la plataforma para dispositivos móviles Android y emplea el kernel de Linux para realizar

tareas a bajo nivel. Es posible escribir aplicaciones en C/C++ para ejecutarlas directamente en el kernel de Linux haciendo uso del Native Development Kit (NDK) que provee Android.

Esta máquina virtual utiliza los ficheros ejecutables Dalvik (`.dex`) que están optimizados para garantizar el mínimo consumo de memoria. En el proceso de compilación, la máquina virtual utiliza los ficheros `.class` generados y los combina en uno o más ficheros `.dex`. Reutiliza información duplicada en múltiples ficheros `.class` para así reducir a la mitad el espacio requerido de un fichero `.jar` tradicional.

2.3.3. Componentes de una aplicación

Los componentes son los bloques esenciales de una aplicación. Cada componente es un punto distinto por el que el sistema puede entrar en nuestra aplicación. No todos los componentes son puntos de entrada para el usuario y algunos de ellos dependen de otros. Sin embargo, cada punto existe por sí solo como una entidad y tiene un papel específico, según se comenta en [1].

Existen cuatro tipos de componentes en una aplicación, que se explican en los siguientes apartados.

Activity

Este componente representa una pantalla con una interfaz de usuario. Sin embargo, las *Activities* trabajan juntas para conseguir cierta continuidad en la experiencia del usuario, a pesar de que son independientes entre ellas.

Normalmente, en una aplicación exiten varias *Activities* que están conectadas entre ellas, y una de ellas es identificada como la principal que será la que se presente al usuario al lanzar la aplicación. Cada vez que una *Activity* es lanzada, la anterior se para pero el sistema la almacena en una pila llamada *Back Stack* que sigue una política de servir primero el último que llega a la cola (Last in First out, LIFO). De modo que, cuando un usuario pulsa el botón atrás del teléfono, la *Activity* actual se destruye, notificando de este cambio a través de los métodos de *callback* del ciclo de vida de la *Activity*, y la anterior se reanuda. En la figura 2.21 se ilustra el ciclo de vida de una *Activity*.

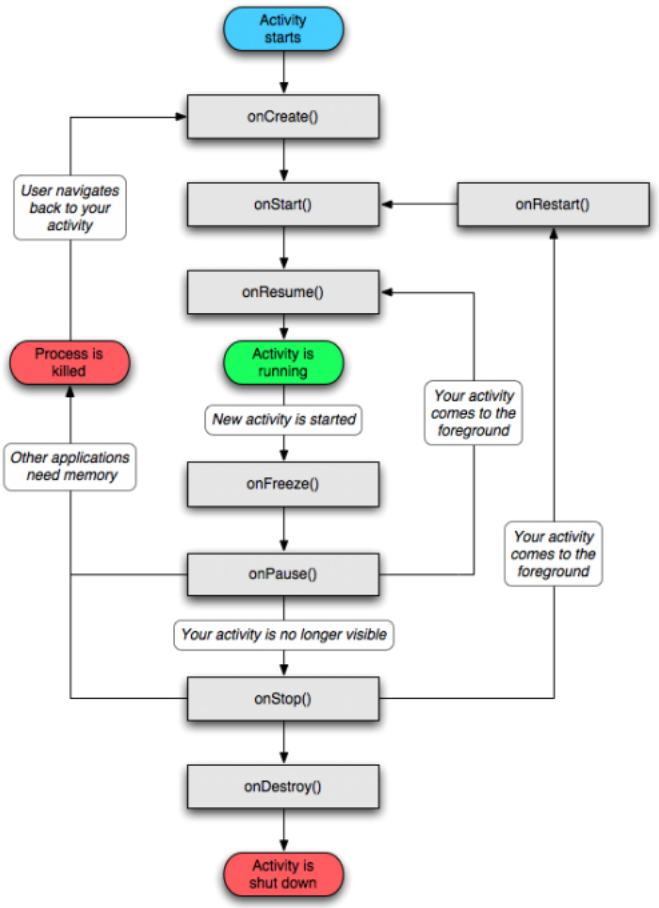


Figura 2.21: El ciclo de vida de una *Activity*, tomado de [1]

Una *Activity* puede existir en tres estados:

- *Resumed*: la *Activity* está en primer plano y el foco del usuario está sobre ella.

- *Paused*: otra *Activity* está en primer plano y tiene el foco del usuario, pero nuestra *Activity* todavía es visible. Es decir, la otra *Activity* está situada encima de la nuestra pero es parcialmente transparente o no cubre toda la pantalla.

- *Stopped*: nuestra *Activity* está completamente oculta tras otra Activity situada en primer plano, pero sigue existiendo.

Service

Se trata de un componente que se ejecuta en segundo plano para realizar operaciones de larga duración. Este componente no proporciona una interfaz de usuario. Otros componentes, como las *Activities*, permiten lanzar *Services* y conectarse a ellos cuando el momento lo requiera o incluso realizar comunicaciones entre procesos (Interprocess Communication, IPC).

Los estados de un servicio se resumen en dos:

- *Started*: cuando el *Service* es iniciado por otro componente llamando al método `startService()`. Una vez iniciado, un *Service* puede estar ejecutándose en segundo plano indefinidamente aunque el componente que lo inició sea destruido.

- *Bounded*: un componente se conecta al *Service* con la llamada al método `bind()`, y en el caso de que el servicio no estuviese lanzado, también se iniciaría. De modo que, el *Service* ofrece una interfaz cliente-servidor que permite a los componentes interactuar con él, enviar peticiones, obtener resultados, etc. Sin embargo, si el componente que se conectó, es destruido, el *Service* también será destruido.

Content Provider

Un Content Provider se encarga de compartir un conjunto de datos de la aplicación. Podemos almacenar estos datos en el sistema de ficheros, en una base de datos, en una página web, o en cualquier otra localización de almacenamiento persistente a la que la aplicación pueda acceder.

A través del *Content Provider* otras aplicaciones pueden acceder o incluso modificar estos datos. Además, también son útiles para escribir y leer información que es privada en nuestra aplicación. También permite hacer público contenido propio, bien creando un *Content Provider* personalizado, o bien añadiendo la información a uno existente.

Broadcast Intent Receiver

Se trata de un componente que responde a avisos emitidos por el sistema y está pensado para no realizar un trabajo pesado. Igualmente, las aplicaciones también pueden emitir avisos. A pesar de que el *Broadcast Receiver* no puede mostrar información al usuario, puede mostrar notificaciones en la barra de estado del teléfono para comunicar al usuario de un evento.

De acuerdo con [15] existen dos tipos de mensajes *broadcast*: ordenados y no ordenados. En el modo no ordenado, los mensajes se envían a todos los

receptores interesados al mismo tiempo. Esto quiere decir que un receptor no puede interferir en la labor de otros receptores y tampoco puede evitar que otros receptores se ejecuten. Un ejemplo de este modo es el mensaje de nivel de batería bajo.

En el modo ordenado, los mensajes broadcast se envían a cada receptor en un orden controlado por el atributo `android:priority` del manifiesto, y en este caso, un receptor es capaz de abortar un mensaje *broadcast*, de modo que, los receptores con menor prioridad no lo recibirán. Un ejemplo de este modo es la notificación de un mensaje de texto entrante.

2.3.4. Política de eliminación de procesos en Android

Uno de los aspectos más importantes de Android es que cualquier aplicación puede iniciar componentes de otra aplicación. Según [3], cuando el sistema inicia un componente significa que inicia el proceso para esa aplicación, a no ser que ya estuviese iniciado, e instancia las clases necesarias para dicho componente.

Como el sistema lanza cada aplicación en procesos distintos con permisos de fichero que restringen el acceso a otras aplicaciones, nuestra aplicación no puede activar directamente un componente de otra aplicación. Para ello, debemos enviar un mensaje al sistema para que él lo inicie por nosotros.

Las aplicaciones en Android confían en un manejo automático de la memoria por parte del recolector de basura de Dalvik, lo que algunas veces puede afectar al funcionamiento de la aplicación si no se tiene especial cuidado con las reservas de memoria.

Durante el ciclo de vida de una *Activity*, el proceso podrá ser eliminado cuando el punto de ejecución de la *Activity* se encuentre en los métodos `onPause()` o `onStop()`, es decir, cuando no tiene el foco de la aplicación. Puede ocurrir que Android elimine un proceso al cual el usuario quiera volver pulsando el botón atrás del teléfono, en este caso, dicho proceso se restaura gracias a una copia y vuelve a estar activo como si no se hubiese eliminado. Existen cinco tipos de procesos en Android que quedan recogidos en la figura 2.22.

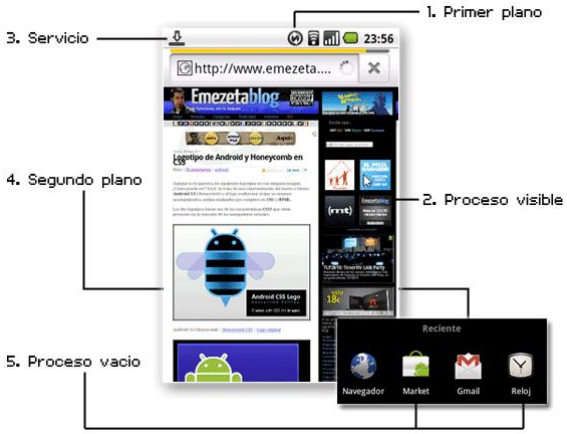


Figura 2.22: Ejemplo de la ejecución de varios procesos en un teléfono Android

- Procesos de primer plano: son aquellos procesos pertenecientes a una tarea que se está ejecutando en ese momento, por ejemplo, revisar el correo.
- Procesos visibles: se puede visualizar la tarea en la pantalla, por ejemplo, navegar por la página.
- Procesos de servicio: se tratan de procesos pendientes de alguna tarea, por ejemplo, descargar un archivo mp3.
- Procesos de segundo plano: son tareas interrumpidas que pueden ser reanudadas si el usuario lo desea, por ejemplo, el cliente de Gmail o Twitter.
- Procesos vacíos: son tareas que se suponen finalizadas, pero que se mantienen porque se considera que pueden ser utilizadas en breve, por ejemplo, el Android Market o el reloj.

Según se comenta en [9], una vez haya muchas reservas de memoria, el recolector de basura pasará y liberará parte de la memoria. En la mayoría de las veces, el recolector de basura pasa lo suficientemente rápido como para que el usuario no lo detecte. Sin embargo, si el recolector de basura se ejecuta mientras el usuario se está desplazando por una lista de elementos, la capacidad de respuesta de la aplicación se ve afectada y ralentizada. Normalmente, el recolector de basura se ejecuta debido a la existencia de objetos de corto ciclo de vida.

Para detectar reservas de memoria en las aplicaciones Android existe una herramienta llamada *Allocation Tracker* (en [9] se explica cómo debe ser utilizada). En el momento que Android detecta que no hay recursos suficientes en el teléfono, elimina aquellos procesos que sean menos prioritarios, y así libera recursos.

Por otra parte, las aplicaciones en Android tienen una memoria dinámica (Heap Memory, HM) limitada, que para algunas aplicaciones, más en concreto las basadas en localización y posicionamiento, puede no llegar a ser suficiente, y en el caso de que no se vaya a utilizar toda la memoria, se debería emplear lo menos posible. Cuantas más aplicaciones puedan quedarse en memoria, más rápido para el usuario es navegar entre ellas.

En la mayoría de los casos, cuando se producen pérdidas en memoria es debido a mantener referencias a los contextos de las *Activities*. Estos contextos se emplean para muchas operaciones, pero normalmente se utilizan para cargar y acceder a recursos. Este es el motivo por el que todos los componentes reciben un contexto en el constructor. En Android existen dos tipos de contextos, el de las *Activities* y el de la aplicación. Habitualmente, el primero es el que se suele utilizar en las llamadas a métodos que necesiten un contexto para su ejecución. De modo que, si el contexto de una *Activity* está siendo utilizado por otros objetos, a pesar de que dicha *Activity* deba ser destruida, no podrá destruirse hasta que el objeto que está utilizando su contexto lo libere.

2.3.5. Uso de múltiples hilos en Android

Cuando una aplicación es lanzada, el sistema crea un hilo principal llamado *UI Thread* responsable de enviar eventos a los componentes adecuados, incluyendo los eventos de pintado. Este hilo también se encarga de interactuar con los componentes que se estén ejecutando.

Este modelo de un único hilo puede llevar a un mal rendimiento a no ser que nuestra aplicación se implemente de forma correcta. Especialmente, si todo se ejecuta en un solo hilo, realizar operaciones pesadas como puede ser acceder a la red o a la base de datos puede bloquear la interfaz de usuario. En el caso de que tengamos que realizar operaciones que puedan llevar bastante tiempo, debemos asegurarnos de que se realizan en otros hilos.

Según [8] Android ofrece muchas formas de acceder al hilo principal desde otros hilos:

- `Activity.runOnUiThread(Runnable)`.
- `View.post(Runnable)`.

- `View.postDelayed(Runnable,long)`.
- `Handler`.

Desafortunadamente, estas clases y estos métodos podrían también aumentar la complejidad del código y convertirlo en ilegible.

Una posible solución a estos problemas es la clase conocida como `Asynctask` que simplifica la creación de tareas que conlleven mucho tiempo y que necesitan comunicarse con la interfaz de usuario. Para utilizarla debemos crear una subclase de este tipo, que debe ser instanciada en el hilo principal y solo puede ejecutarse una vez. Además, puede ser cancelada desde cualquier hilo en cualquier instante. El método `doInBackground()` se ejecuta en un *Worker Thread*. Los métodos `onPreExecute()`, `onPostExecute()` y `onProgressUpdate()` son llamados por el hilo principal.

2.3.6. Seguridad en Android

Android es un sistema de privilegios separados, es decir, cada aplicación se ejecuta con un identificador de sistema distinto. De acuerdo con [2], el punto clave de la seguridad en Android es que ninguna aplicación, por defecto, tiene permisos para realizar ninguna operación que pueda impactar de forma negativa en otras aplicaciones o en el sistema operativo. Android proporciona a cada paquete una identificación de usuario Linux distinta. Esta identidad se mantiene constante durante el ciclo de vida de dicho paquete en el teléfono.

Las aplicaciones deben declarar los permisos que necesitan para utilizar capacidades que no se abarcan dentro del paquete básico. De tal modo que, Android muestra en el momento de la instalación una ventana al usuario advirtiendo de los permisos que necesita la aplicación para funcionar, que deberán ser aceptados por el usuario para que la instalación se realice correctamente.

2.3.7. APIs de Android empleadas en las aplicaciones basadas en posicionamiento y localización

Este apartado tiene como objetivo hacer una breve introducción a los paquetes de Android que serán empleados en el desarrollo de nuestra aplicación. Existen dos tipos de APIs: internas y externas, que a continuación se describen.

Librerías internas de Android

- Almacenamiento persistente en Android

Existen múltiples formas de almacenar información en una aplicación de forma persistente según [4]: en las preferencias, en la memoria interna o compartida del teléfono, en una base de datos o en la red.

- Android content. Nos permite almacenar información tanto en la memoria interna o compartida del teléfono, permitiéndonos crear archivos privados o compartidos que sean accesibles por la aplicaicón o también por otras aplicaciones.

  En el caso de que nos interese guardar en caché información no persistente, Android facilita el método `getCacheDir()` para abrir un fichero que representa el directorio interno donde la aplicación debe guardar temporalmente los archivos.

- Android SQLite. Android proporciona un API que contiene clases que gestionan la creación de bases de datos SQL y la ejecución de sentencias SQL. Las aplicaciones pueden utilizar dichas clases para gestionar bases de datos privadas. Cualquier base de datos será accesible desde todos los componentes de la aplicación, pero no desde fuera de la misma según [4].

  Para escribir o leer de la base de datos, se debe llamar a los métodos `getWritableDatabase()` y `getReadableDatabase()`, respectivamente, que devuelven un objeto de tipo `SQLiteDatabase` que representa la base de datos y proporciona métodos para realizar operaciones SQL.

- Android net. Este API facilita las clases necesarias para establecer conexiones a través de la red, como puede ser, cliente HTTP, gestionar *streamming* mediante RTSP, mantener sesiones SIP y manejar las funcionalidades de conexiones WiFi.

- Android Location

  Para el desarrollo de aplicaciones que necesiten conocer la posición del usuario, Android permite el acceso al GPS del dispositivo y al proveedor de red. Aunque el GPS es la tecnología más precisa hasta el momento, sólo funciona correctamente en exteriores, consume mucha batería, y además tarda mucho en obtener la posición del usuario por primera vez. Sin embargo, el proveedor de red de localización (`Network Location Provider`, NLP) de Android utiliza métodos que emplean las señales de las estaciones base y de los puntos de acceso WiFi para proporcionar información sobre la localización del usuario, y funciona tanto en interiores como exteriores, según se comenta en [7], además de responder de forma más rápida y consumir menos batería.

Obtener la posición del usuario puede ser realmente complicado debido a la presencia de posibles fuentes de error, como pueden ser:

- Múltiples fuentes de localización: GPS, Cell-ID y WiFi pueden proporcionarnos información de localización de usuario, sin embargo, es difícil determinar cual debe utilizarse teniendo en cuenta la velocidad de respuesta, precisión y consumo de batería requeridos por la aplicación a desarrollar.
- Movimiento del usuario: se debe tener en cuenta que el usuario se va desplazando, y por tanto, debe re-estimarse su posición cada cierto tiempo.
- Variación de la precisión: la localización obtenida con una fuente puede ser más precisa que la localización obtenida después por otra fuente, es decir, la precisión varía en función de la metodología de posicionamiento utilizada.

- Android Telephony

  El API de telefonía de Android se utiliza, entre otras cosas, para monitorizar información del teléfono móvil como puede ser su estado, sus conexiones o incluso sus mensajes de texto.

  Para comprobar el estado del teléfono, el estado del servicio, el nivel de señal o el indicador de señal, se debe implementar la clase `PhoneStateListener` y asociarlo a un objeto de la clase `TelephonyManager`. Dicha clase proporciona acceso a información sobre los servicios del teléfono como puede ser el operador, información de las celdas vecinas y de la celda a la que está conectado, el tipo de red a la que está conectado el móvil, la versión del *software* del teléfono e información sobre la SIM si el operador lo permite. Las aplicaciones pueden utilizarla para determinar los servicios y los estados de un teléfono. Parte de esta información es protegida por permisos, y para acceder a ella habría que definir los permisos pertinentes en el manifiesto de la aplicación.

Librerías Externas de Android

Las librerías externas no forman parte de las librerías estándar de Android, y por tanto, pueden no estar presentes en algunos dispositivos Android.

- Google Maps API

Como se ha mencionado en el apartado anterior, para dar acceso a los datos de localización a las aplicaciones, Android proporciona la API de localización, que junto con la librería externa de Google Maps nos permitirá desarrollar una aplicación de localización basada en mapas, según se comenta en [6].

La clase `MapView` del paquete `google.android.maps` muestra un mapa con datos obtenidos del servicio de Google Maps. Esta vista puede capturar los eventos táctiles y de los botones del teléfono para aplicar *zoom* sobre los mapas y proporciona todos los elementos necesarios para que el usuario interactúe con ellos.

Para poder mostrar una vista en nuestra aplicación de Google Maps necesitamos obtener una clave que nos de permisos. Ver anexo C

- Geolocation API

  De acuerdo con [5], proporciona la mejor estimación de la posición del usuario utilizando varios proveedores de localización. Estos proveedores pueden ser tales como GPS o proveedores de red.

  Una vez obtengamos el identificador de celda, el código de área de localización (Location Area Code, LAC), el código móvil del país (Mobile Country Code, MCC) y el código móvil de la red (Mobile Network Code, MNC), se lanza una petición HTTP de tipo POST con estos datos y el servidor nos devolverá la información de localización de la torre correspondiente a dicha celda en términos de longitud y latitud.

Capítulo 3

Desarrollo de DroidTriangulation

El objetivo de este capítulo es explicar detalladamente el contenido de las partes que forman la aplicación y destacar las funconalidades más relevantes que nos proporciona la plataforma Android para llevar a cabo nuestro objetivo de posicionar a un usuario conociendo la posición de tres estaciones base y comparar su precisión respecto a la localización que nos proporciona el GPS instalado en el dispositivo móvil y la localización basada en redes mediante la información que Google proporciona.

3.1. Introducción a DroidTriangulation

Esta aplicación permite mostrar al usuario su posición y la localización de las tres estaciones base de las cuales el terminal recibe mayor potencia y que se emplean para realizar los cálculos de los cuatro centros que se nombraron en el capítulo 2 en la sección donde se describían los cuatro centros escogidos 2.2.1.

El objetivo de la aplicación es mostrar dichas posiciones y almacenar información sobre ellas para después realizar un posterior análisis comparándolo con la localización obtenida por un dispositivo GPS autónomo, además de evaluar las posibles prestaciones de la aplicación.

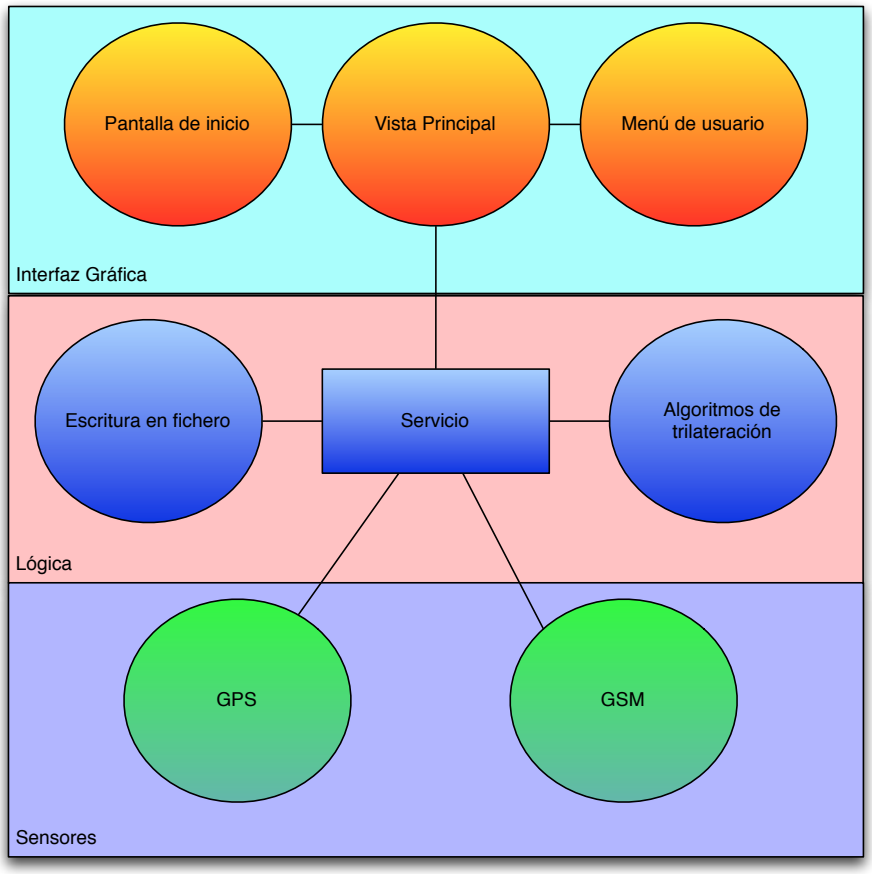


Figura 3.1: Estructura de la aplicación

Esta aplicación puede dividirse principalmente en tres capas tal y como se refleja en la figura 3.1:

- Capa de Sensores. Este nivel de la arquitectura es el que se encarga de recoger la información que proviene de los sensores del dispositivo móvil.
- Capa lógica. En esta capa la aplicación tiene un servicio Android que se encarga de procesar los datos que provienen de los sensores para calcular la posición del usuario y una vez calculada almacenar toda la información en ficheros.
- Capa interfaz gráfica. Este nivel compone las vistas de la aplicación que permiten al usuario conocer su posición sobre un mapa y escoger el tipo de localización que desee.

3.1.1. Sensores

Tal y como se muestra en la figura 3.1 esta capa recoge información que proviene del dispositivo GPS integrado en el móvil y permite configurar la frecuencia de actualización de la posición del usuario. Además, cuando el dispositivo móvil detecta un cambio de celda, se encarga de obtener la información de la estación base a la cual está conectado el usuario.

3.1.2. Lógica

En segundo plano se calculan los cuatro centros del triángulo formado por las tres estaciones base. Para ello, tras obtener la información de la estación base a la que estamos conectados podemos pedir al API de Google su localización, al igual se hace con las dos estaciones vecinas y una vez tenemos la latitud y la longitud de estas tres estaciones podemos aplicar los métodos de trilateración mencionados anteriormente en la sección 2.2.1.

Para cada actualización de la posición del usuario, bien sea procedente del GPS o de un cambio en la estación base a la que estamos conectados, se guarda información en fichero sobre la localización del usuario, sobre las estaciones base que se han empleado en los cálculos y el tiempo que se ha tardado en determinar la posición.

Estas operaciones se hacen en paralelo de forma que cuando se obtengan sus coordenadas se muestran en la pantalla principal de la aplicación ante el usuario con diferentes colores para poder distinguirlos.

Cabe destacar que para acceder a la información que guarda en fichero la aplicación el móvil se debe disponer de una tarjeta de almacenamiento SD en la que al cerrar la aplicación se volcará dicha información recogida en varios ficheros.

3.1.3. Interfaz Gráfica

Al lanzar la aplicación se muestra la pantalla de inicio que aparece en la figura 3.2, mientras se realizan las tareas de configuración principales de la aplicación, que una vez finalizadas, nos redirigen a la vista principal.

Figura 3.2: Pantalla de inicio

Cuando iniciamos la aplicación por primera vez tardará varios segundos en determinar la posición del usuario hasta que el GPS obtenga señal, sin embargo una vez la aplicación nos localice por primera vez se quedará almacenada la última posición que el usuario obtuvo en la última ejecución de la aplicación.

La pantalla principal muestra una vista perteneciente a Google Maps y se superponen iconos Android de diferentes colores. Adicionalmente, el usuario puede hacer *zoom* con los controles que aparecen en la parte inferior de la pantalla y en la parte superior se mostrará el nombre de la calle en la que el GPS ha situado al usuario, tal y como se puede ver en la figura 3.3. Puede darse la ocasión que en esta barra superior a veces no aparezca el nombre de la calle, esto se debe a que para la localización obtenida del usuario Google no tenga registrada una dirección con el nombre de la calle.

La correspondencia entre los colores y las posiciones calculadas es la siguiente.

- GPS o localización por red: verde
- Incentro: amarillo
- Circuncentro: rojo
- Ortocentro: rosa
- Centroide: azul

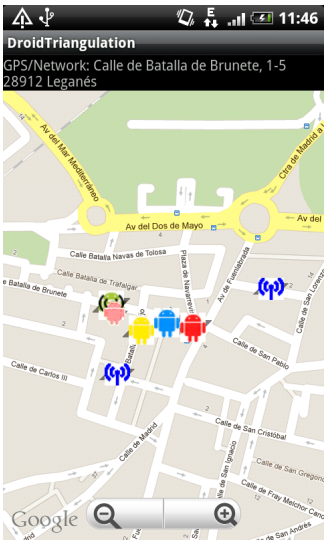


Figura 3.3: Vista principal de la aplicación

- Dos estaciones base vecinas: antenas azules
- Estación base a la que el dispositivo móvil está conectado: antena negra.

La aplicación dispone de un menú de usuario que aparece en la parte inferior de la pantalla, tal y como se muestra en la figura 3.4 tras pulsar el botón "menú" del dispositivo android dando a elegir al usuario entre las siguientes opciones:

- Salir. Cerrar la aplicación.
- Preferencias. Permite al usuario consultar su posición en función de cada uno de los centros calculados a través de métodos de lateración y la posición GPS. Y por otra parte, el usuario puede escoger el tipo de localización que quiere utilizar: GPS, GSM o ambas simultáneamente.
- Buscar redes WiFi. Permite realizar un barrido de las redes inalámbricas que detecta el dispositivo móvil y almacena dicha información en un fichero.

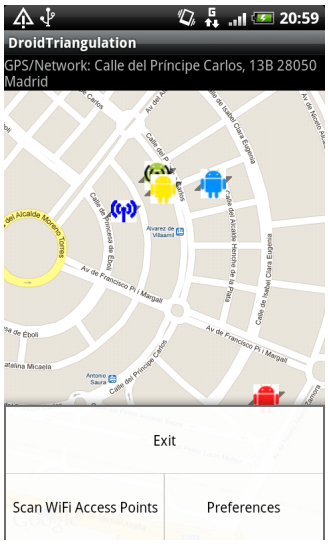


Figura 3.4: Vista principal de la aplicación con el menú desplegado

3.2. Detalles de implementación

En este apartado se explicará brevemente la funcionalidad de cada una de las clases que componen la aplicación y se mostrará un esquema especifico de cada una de ellas y finalmente un esquema general de toda la aplicación.

Tal y como se mencionaba anteriormente, la aplicación puede dividirse en tres capas, de tal forma que la estructura de la aplicación por mantener la coherencia se ha dividido en diferentes paquetes que recogen cada una de las funcionalidades mostradas en la figura 3.1. La figura3.5 muestra la correspondencia entre los paquetes de la aplicación y las capas mencionadas anteriormente:

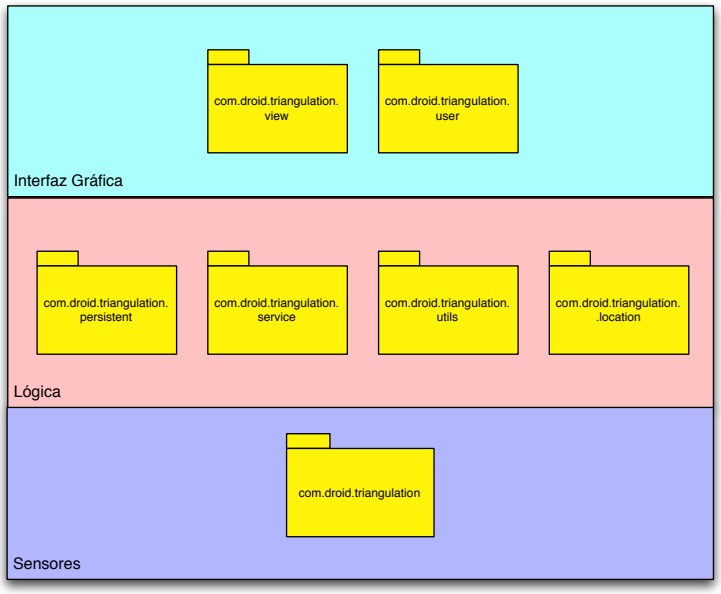


Figura 3.5: Estructura de los paquetes de la aplicación

3.2.1. `com.droid.triangulation`

En este paquete se centraliza el control de datos que obtenemos a partir de los sensores del dispositivo móvil.

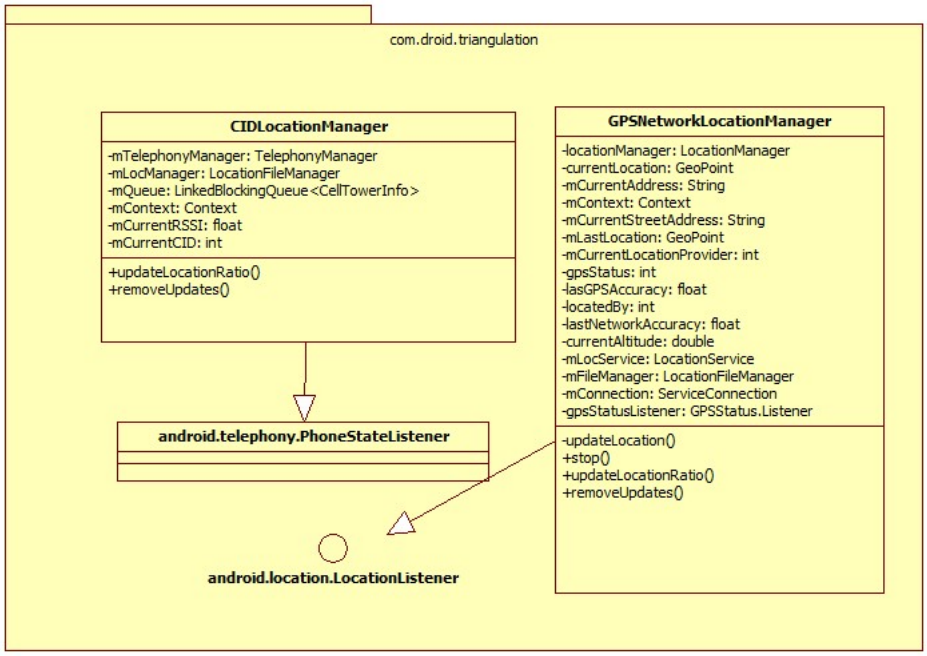


Figura 3.6: Paquete `com.droid.triangulation`

- `CIDLocationManager`: clase que se encarga de escuchar los cambios de celda del dispositivo móvil, almacenando la información de la celda a la que estamos conectados y de las celdas vecinas en una cola de planificación para su posterior procesado.
- `GPSNetworkLocationManager`: clase que dependiendo de la frecuencia de actualización que fijemos, actualiza la información referente a la posición del usuario según el GPS del dispositivo móvil.

3.2.2. `com.droid.triangulation.location`

Paquete que se encarga de calcular las posiciones mediante métodos de trilateración y que proporciona una entidad para guardar la información referente a la estación base a la que estamos conectados.

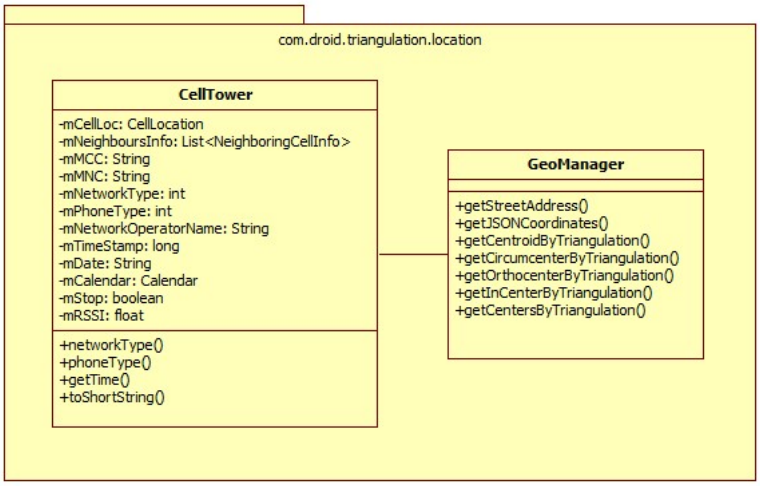


Figura 3.7: Paquete `com.droid.triangulation.location`

- `CellTower`: entidad destinada a almacenar la información referente a la estación base a la que el dispositivo móvil está conectado.
- `GeoManager`: clase que contiene los métodos de lateración para el cálculo de cada uno de los centros (incentro, centroide, circuncentro, ortocentro) y el método de conexión con la API de Google para pedir las coordenadas de las estaciones base en función de su CID, su LAC, su MCC y su MNC.

3.2.3. `com.droid.triangulation.service`

Este paquete contiene el servicio que se encuentra ejecutándose en segundo plano en la aplicación.

- `LocationService`: clase que se ejecuta en segundo plano desde que se abre la aplicación hasta que se cierra y que se encarga de procesar la información proporcionada por las clases `GPSNetworkLocationManager` y `CellIDLocationManager` y notificar a la `Activity` principal de las nuevas posiciones calculadas.

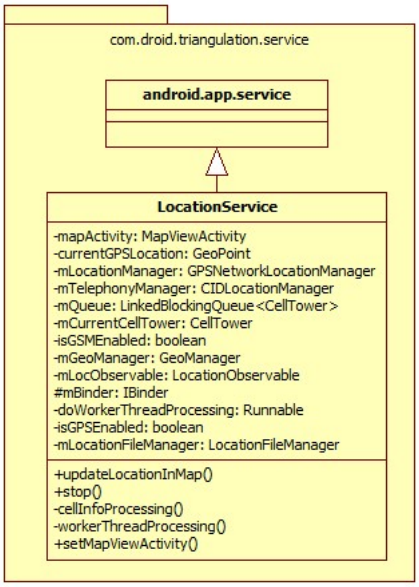


Figura 3.8: Paquete `com.droid.triangulation.service`

3.2.4. `com.droid.triangulation.view`

Este paquete engloba las clases responsables de mostrar las localizaciones calculadas en una vista de un mapa y la pantalla de inicio de la aplicación.

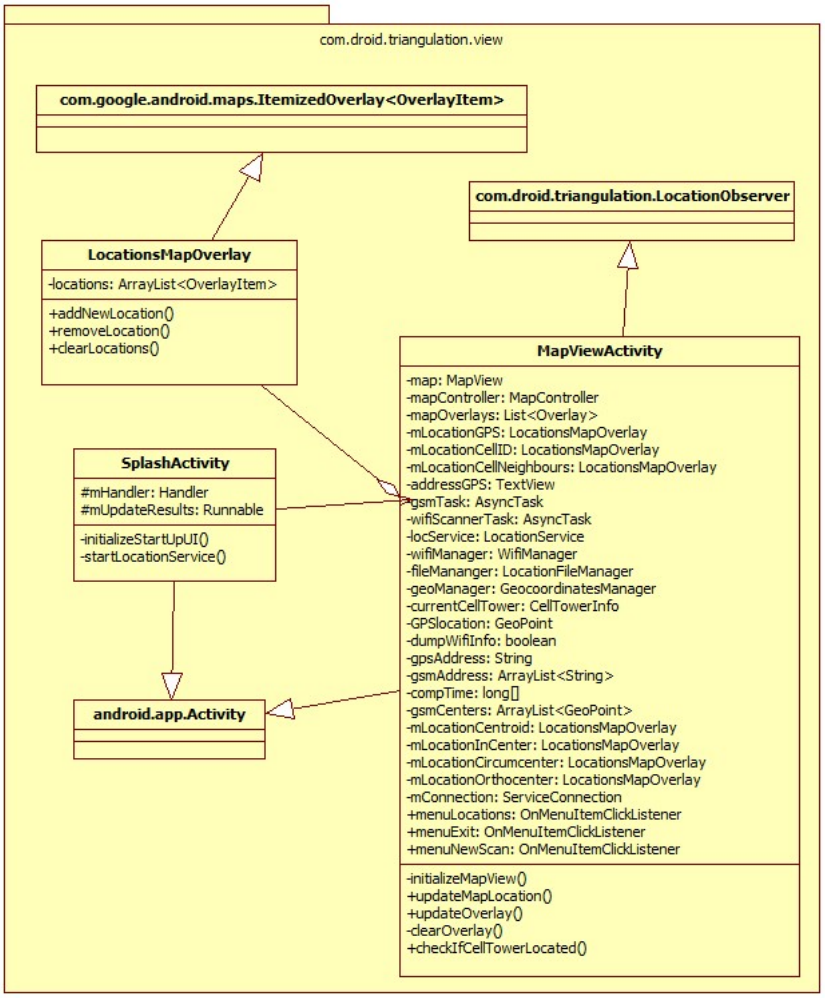


Figura 3.9: Paquete `com.droid.triangulation.view`

- `SplashActivity`: clase que compone la pantalla inicial de la aplicación, que muestra una imagen mientras se realizan una serie de operaciones.

- `LocationsMapOverlay`: proporciona una capa que permite añadir o eliminar localizaciones a dicha capa.

- `MapViewActivity`: es la clase que representa la vista principal de la aplicación, que en función de las opciones activadas por el usuario mostrará las localizaciones calculadas mediante iconos diferentes para que sean distinguidas visualmente.

3.2.5. `com.droid.triangulation.persistent`

Paquete que contiene la clase que se encarga de guardar en fichero los datos recogidos por la aplicación.

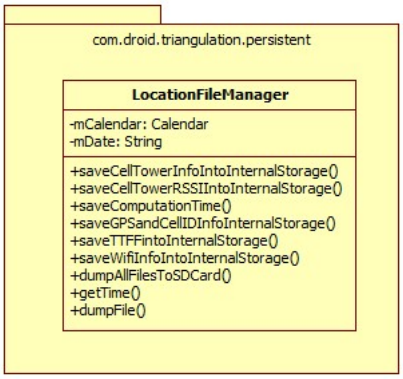


Figura 3.10: Paquete `com.droid.triangulation.persistent`

- `LocationFileManager`: clase que se encarga de guardar en fichero los datos recogidoa por la aplicación: los distintos puntos de localización, información sobre las estaciones base, etc.

3.2.6. `com.droid.triangulation.properties`

Este paquete recoge un fichero que contiene todos los *logs* que muestran información durante el ciclo de vida de la aplicación.

3.2.7. `com.droid.triangulation.user`

Paquete que contiene el archivo de preferencias de la aplicación en el que se guarda la localización a utilizar para posicionar al usuario, `Preferences`. Además, permite conocer el la localización calculada para cada una de las posiciones del usuario.

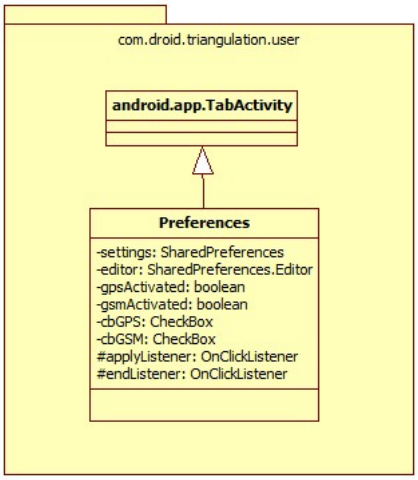


Figura 3.11: Paquete `com.droid.triangulation.user`

3.2.8. `com.droid.triangulation.utils`

Este paque contiene las clases `LocationObserver` y `LocationObservable` que componen un patrón *Observer* empleado para actualizar la interfaz de la aplicación cuando se recibe un cambio en la posición del usuario.

La figura 3.12 muestra un resumen de las distintas clases que componen la aplicación y la relación existente entre ellas.

3.3. Interfaz gráfica

Como se comentó en la sección 2.3, cada una de las pantallas que componen una aplicación android está contenida en un elemento conocido como *Activity*.

La aplicación está formada por la pantalla principal con la vista de un mapa de Google sobre el que se representan las localizaciones calculadas (`MapViewActivity`) y el menú de configuración de la aplicación (`Preferences`). A continuación se explicará brevemente las funcionalidades de cada una de estas vistas.

3.3.1. `MapViewActivity`

Esta clase se encarga de mostrar las localizaciones procedentes del GPS y las calculadas mediante métodos de lateración.

Para ello se encuentra conectada al servicio en segundo plano responsable de escuchar las actualizaciones de localización procedentes de los sensores del dispositivo móvil, de tal forma que, cada vez que se produce una actualización en la localización del usuario, lo procesa y se lo comunica a esta *Activity* para que lo refleje en el mapa. El siguiente código muestra cómo conectarse a un servicio:

```
1
2  private ServiceConnection mConnection = new
       ServiceConnection() {
3  public void onServiceConnected(ComponentName className,
       IBinder service) {
4          locService = ((LocationService.LocalBinder)
               service).getService();
5          locService.setMapViewActivity(MapViewActivity.
               this);}};
6  @Override
7  public void onCreate(Bundle savedInstance){
8          super.onCreate(savedInstance);
9          Intent intent = new Intent(this,LocationService.
               class);
10         if(intent!=null && mConnection!=null){
11                 boolean connected = this.bindService(
                       intent, mConnection,Context.
                       BIND_AUTO_CREATE);}}
```

La vista del mapa es un elemento nativo llamado `MapView`, y se puede

componer por varias capas, que actualmente suelen emplearse para dar la opción al usuario de mostrar lo puntos de interés que él desee (hospitales, bares, centros comerciales, etc.). Sin embargo, en nuestro caso no se le proporciona al usuario esta posibilidad puesto que lo que nos interesa es que se muestren al mismo tiempo todas las localizaciones calculadas para que el usuario sea consciente la precicisón de cada uno de los métodos empleados. Estas capas se crean haciendo uso de la clase nativa `Overlay`, a las que se le irán añadiendo las localizaciones.

Para mostrar cada una de las posiciones calculadas, se hace uso de la clase `LocationsMapOverlay` que hereda de la clase nativa de Android `ItemizedOverlay<OverlayItem>` y que cuya finalidad es simular marcadores sobre la vista de un mapa. Por lo tanto, tendremos un objeto `LocationsMapOverlay` por cada posición que queramos representar que se distinga por un icono distinto, es decir, en el caso de las dos estaciones base vecinas, al mostrarse con el mismo tipo de icono, sólo necesitamos un objeto de este tipo y le añadiremos dos posiciones a través de su método `addNewLocation()`.

A continuación se muestra un extracto del código necesario para visualizar posiciones sobre un mapa:

```
1
2  map = (MapView)findViewById(R.id.mapview);
3  mapOverlays = map.getOverlays();
4  myCurrentLocationGPS = new LocationsMapOverlay(
      getResources().getDrawable(R.drawable.droid_marker));
5          if(GPSlocation==null){
6                  myCurrentLocationGPS.addNewLocation(point
                      , Constants.EMPTY_STRING, Constants.
                      EMPTY_STRING);
7          }else{
8                  myCurrentLocationGPS.addNewLocation(
                      GPSlocation, Constants.EMPTY_STRING,
                      Constants.EMPTY_STRING);
9                  mapController.setCenter(GPSlocation);
10         }
11 mapOverlays.add(myCurrentLocationGPS);
```

Puede ocurrir el caso de que las posiciones calculadas por cada uno de los métodos se encuentren muy separadas en distancia y, por tanto, no se visualicen todas en la pantalla del dispositivo. Por ello, se proporcionan unos controles con los que el usuario puede ampliar o reducir el *zoom* sobre el mapa.

En lo referente al menú que se le presenta al usuario, esta *Activity* es la que se encarga de implementar esta funcionalidad. En nuestro caso, hemos empleado la tecla menú de los dispositivos móviles, lo que significa que nuestra *Activity* simplemente debe sobreescribir el método de *callback* `onCreateOptionsMenu()` y registrar un esuchador `onMenuItemClick()` distinto para cada una de las opciones. El siguiente código muestra un ejemplo de como debe implementarse este método para mostrar un menú con tres opciones : "Exit", "Scan WiFi Access Points" y "Preferences"

```
1
2 @Override
3 public boolean onCreateOptionsMenu(Menu menu){
4         super.onCreateOptionsMenu(menu);
5         MenuItem menuItem = menu.add(groupId, menuItemId,
              menuItemOrder, R.string.menu_exit);
6         menuItem.setOnMenuItemClickListener(menuExit);}
```

Finalmente, para no sobrecargar al hilo que se encarga de la interfaz de la aplicación, las tareas de barrido de redes inalámbricas y guardar la información de localización del usuario se realizan en segundo plano haciendo uso de la clase `AsyncTask` que comentábamos en la sección 2.3.

3.3.2. `Preferences`

Existe una *Activity* especial llamada `PreferenceActivity` que nos proporciona la facilidad de crear elementos visuales con la lógica ya implementada para que sus valores se almacenen de forma automática en las preferencias de la aplicación. Sin embargo, en nuestro caso hemos empleado una `TabActivity` porque no sólo una de las pestañas está destinada a escoger el tipo de localización a utilizar, sino que la otra pestaña simplemente muestra información al usuario sobre su localización y el formato de este componente de Android es más adecuado para ello.

Esta clase hereda de un tipo especial de la clase `TabActivity` que será la encargada de crear todas las pestañas que queramos, nos permitirá crear una clase del tipo que nos interese con su correspondiente interfaz por cada pestaña que queramos introducir. Este sistema tiene muchas ventajas como pueden ser tener el código más dividido y limpio, que una pestaña pueda ser una lista y otra una simple *Activity*, que cada pestaña tenga su propio menú, etc. Las vistas correspondientes a cada pestaña son añadidas a un elemento `TabHost` que nos proporciona este tipo de *Activities*.

A continuación se muestra cómo debe de ser inicializada una

`TabActivity`:

```
1
2 TabHost mTabHost = getTabHost();
3 mTabHost.addTab(mTabHost.newTabSpec(TAB1).setIndicator(
    PREFERENCES,getResources().getDrawable(android.R.
    drawable.ic_menu_mapmode)).setContent(R.id.linear_tab1
    ));
```

3.4. Funcionalidades

A continuación, se van a describir las funcionalidades más importantes de la aplicación detalladamente.

3.4.1. Localización por GPS

Para poder proporcionar la posición del usuario, nuestra aplicación debe escuchar los datos proporcionados por los sensores del dispositivo móvil. En nuestro caso, dependiendo de la precisión de dichos datos, en cuanto a términos de distancia respecto a la posisición del usuario, utilizaremos la localización proporcionada por el GPS o por las redes inalámbricas. Este tipo de localización sólo funcionará en el caso de que el usuario tenga activadas las opciones "Utilzar satélites" y "Utilizar redes inalámbricas".

Esta API nos permite obtener la posición del usuario a través de métodos de *callback*. Primero debemos indicar que queremos recibir actualizaciones de localización mediante una instancia de la clase `LocationManager` llamando al método `requestLocationUpdates()` y pasándole como parámetro un objeto de tipo `LocationListener`. Nuestra clase deberá implementar los métodos de dicha interfaz que serán llamados por nuestro `LocationManager` cuando el usuario cambie su posición o el estado del servicio cambie.

Según [7], las siguientes líneas de código muestran un ejemplo de cómo definir un `LocationListener`:

```
1
2 LocationManager locationManager = (LocationManager) this.
    getSystemService(Context.LOCATION_SERVICE);
3 LocationListener locationListener = new LocationListener
    () {
4         public void onLocationChanged(Location location)
            {
```

```
5       makeUseOfNewLocation(location);}
6     public void onStatusChanged(String provider, int status
        , Bundle extras) {}
7     public void onProviderEnabled(String provider) {}
8     public void onProviderDisabled(String provider) {}};
9   locationManager.requestLocationUpdates(LocationManager.
      NETWORK_PROVIDER,0,0,locationListener);
```

Podemos controlar la frecuencia con la que nuestro `LocationListener` recibe las actualizaciones con el segundo y el tercer parámetro del método `requestLocationUpdates()`, inicializándolos a cero significa que reciba actualizaciones lo más frecuentemente posible. Es posible pedir actualizaciones de la localización mediante GPS y mediante el proveedor de red llamando dos veces a dicho método. Para recibir ambas notificaciones por parte de ambos proveedores, se debe habilitar un permiso `ACCESS_FINE_LOCATION` en el manifiesto de la aplicación.

```
1
2   <uses-permission android:name="android.permission.
      ACCESS_FINE_LOCATION"/>
```

En [7] se sugiere un modelo que mejora el comportamiento de las aplicaciones basadas en localización para Android. En nuestro caso la clase responsable de escuchar las actualizaciones de localización GPS o de red de los sensores del dispositivo es `GPSandNetworkLocationManager`.

3.4.2. Localización por red móvil

Por otra parte, para obtener la información sobre la conexión GSM del dispositivo móvil utilizamos el paquete `android.telephony` de la API de Android. Para ello debemos implementar la interfaz `PhoneStateListener` que se compone de un conjunto de métodos *callback* que nos proporcionarán la siguiente información:

- Cambio de celda. Cuando el dispositivo móvil cambie su conexión a una estación base nueva, el dispositivo lanza un evento informando de dicho cambio.
- Cambio de potencia recibida. Cuando la potencia de la estación base a la que el dispositivo está conectado cambia, el dispositivo nos avisa.
- Cambio de estado de llamada. En el ciclo de vida de realización de una llamada, el dispositivo móvil pasa por los siguientes estados:

- *RINGING*. El dispositivo está recibiendo una llamada entrante.
- *OFFHOOK*. El dispositivo móvil ha establecido una llamada.
- *IDLE*. El dispositivo móvil ha finalizado una llamada.

- Cambios en el consumo de datos. Cuando el dispositivo móvil realiza conexiónes a través de la red móvil, podemos recoger información de si se trata de tráfico saliente, o entrante en el móvil.

- Cambios en el estado de la conexión de datos. Cuando se activa o se desactiva la conexión de datos a través de la red móvil, el dispositivo móvil nos avisa de ello.

- Cambios en el estado del servicio del móvil. Podemos detectar cuando el teléfono móvil se encuentra sin cobertura o se está apagando.

En nuestro caso, nos interesa la información referente al cambio de celda y a la variación de la potencia recibida. Para poder recibir dichos eventos debemos registrar nuestro escuchador creando una instantica de la clase `TelephonyManager` y llamando al método `listen()`. El siguiente código muestra un ejemplo de su implementación:

```
1
2  public CIDLocationManager(Context ctx,LinkedBlockingQueue
       <CellTowerInfo> queue) {
3          if(ctx != null){
4                  this.context = ctx;
5                  mTelephonyManager = (TelephonyManager)
                       ctx.getSystemService(Context.
                       TELEPHONY_SERVICE);
6                  updateLocationRatio();}}
7  public void updateLocationRatio() {
8          if(mTelephonyManager!=null){
9                  mTelephonyManager.listen(this,
                       PhoneStateListener.
                       LISTEN_CELL_LOCATION |
                       PhoneStateListener.
                       LISTEN_SIGNAL_STRENGTHS);}}
10 public void removeUpdates(){
11         if(mTelephonyManager!=null){
12                 mTelephonyManager.listen(this,
                       PhoneStateListener.LISTEN_NONE);}}
```

El método `onCellLocationChanged` nos proporciona un objeto de tipo `CellLocation` de cual podremos obtener información acerca de la estación base a la que estamos conectados y de las estaciones base vecinas. Debido a que la frecuencia con la que este método se invoca no la podemos controlar, dicha información la recogemos en un objeto `CellTower` y la añadimos a una cola que se rige por una política de planificación que se basa en servir primero el primer objeto que haya recibido (FIFO, First In First Out). De este modo, el procesamiento se hace en segundo plano sin afectar al hilo principal de pintado de la interfaz.

3.4.3. Obtención de coordenadas de localización

Dentro de la información que recogemos de la estación base se encuentra el MNC, el MCC, el CID y el LAC, que nos permitirá obtener la latitud y longitud de la estación base haciendo una petición a la API de geolocalizacion de Google. El siguiente fragmento de código muestra un ejemplo de dicha petición:

```
{
  "version": "1.1.0",
  "host": "maps.google.com",
  "home_mobile_country_code": 310,
  "home_mobile_network_code": 410,
  "radio_type": "gsm",
  "carrier": "Movistar",
  "request_address": true,
  "address_language": "es_ES",
  "location": {
    "latitude": 51.0,
    "longitude": -0.1
  },
  "cell_towers": [
    {
      "cell_id": "42",
      "location_area_code": 415,
      "mobile_country_code": 310,
      "mobile_network_code": 410,
      "age": 0,
      "signal_strength": -60,
      "timing_advance": 5555
    },
  ]
```

```
25 }
```

También podemos obtener el CID y el LAC de las estaciones base en un objeto del tipo `NeighboringCellInfo` al invocar el método `getNeighborsInfo()` sobre el objeto `CellLocation` que recibimos en el método de *callback* `onCellLocationChanged`. Por lo tanto, esta petición se repetirá para las dos estaciones base de las cuales obtengamos mayor potencia, y así poder calcular nuestra posición una vez conocemos la localización de tres estaciones base.

3.4.4. Información de las estaciones WiFi

La clase `ConnectivityManager` responde a cambios en el estado de conectividad a la red y también notifica a las aplicaciones cuándo la conectividad de la red cambia. Sus responsabilidades son:

- Monitorizar las conexiones de red (WiFi, GPRS, UMTS, etc.).
- Enviar mensajes *broadcast* cuando la conectividad de la red cambia.
- Intentar conectarse a otra red cuando pierde conectividad con la red actual.

El siguiente código muestra un ejemplo para comprobar que se obtiene conectividad a Internet.

```
1
2 public boolean isOnline() {
3         ConnectivityManager cm = (ConnectivityManager)
             getSystemService(Context.CONNECTIVITY_SERVICE)
             ;
4         return cm.getActiveNetworkInfo().
             isConnectedOrConnecting();}
```

Para acceder a dicha información se debe habilitar el siguiente permiso en el manifiesto de la aplicación:

```
1
2 <uses-permission android:name="android.permission.
    ACCESS_NETWORK_STATE"/>
```

En nuestro caso, esta clase se ha empleado para acceder a la información que nos proporciona sobre las redes inalámbricas. De tal modo que, para

guardar la información de la red WiFi a la que estamos conectados basta con tan sólo realizar un barrido. El siguiente código muestra como hacer este proceso:

```
1
2 wifiManager = (WifiManager) getSystemService(Context.
     WIFI_SERVICE);
3 WifiInfo actualWifiInfo = wifiManager.getConnectionInfo()
     ;
4 wifiManager.startScan();
5 results = wifiManager.getScanResults();
```

3.4.5. Almacenamiento persistente

En nuestro caso, nos interesa almacenar los datos de posicionamiento y localización en la memoria interna del teléfono, puesto que no podemos arriesgarnos a almacenarlo en la memoria externa porque puede ocurrir que el usuario no disponga de tarjeta de almacenamiento en su dispositivo móvil, y por tanto, no se guardaría la información de las medidas. Para poder consultar esta información, al salir de la aplicación se vuelca a una tarjeta de almacenamiento externa.

Entre las formas de almacenamiento expuestas en apartado 2.3.7, hemos escogido el almacenamiento en fichero, ya que proporcionando un formato adecuado a los fichero podremos importar los datos de una manera más inmediata a MATLAB para su posterior procesado.

Para ello, Android proporciona el método `openFileOutput()` que devuelve un objeto de tipo `FileOutputStream` que nos permite escribir en fichero y el método `openFileInput()` que devuelve un objeto de tipo `FileInputStream` que nos permite leer de fichero. En nuestro caso tendremos varios ficheros:

- *cellTowerInfo.txt*: almacenamos la información referente a la estación base a la que estamos conectados. Los datos que se guardan siguen este orden: la fecha con foramto yyyymmdd hhmmSSsss, el CID, el LAC, MCC, MNC, el nivel de potencia recibida en dBm, el tipo de red móvil (CDMA, UMTS, etc.) y el nombre de la operadora, tal y como se muestra a continuación.

```
1
2 20120429 121447432       414      2815     214      07
          -101.0  2        movistar
```

- *cellTowerNeighbours.txt*: guardamos información referente a las estaciones base vecinas a la estación base a la que estamos conectados en el siguiente orden: la fecha con foramto yyyymmdd hhmmSSsss, el CID, el LAC, el nivel de potencia recibida en dbm, el PSC y el tipo de red móvil, tal y como se muestra a continuación.

```
1
2 20120429 120757406       611      2815     -89      -1
          2
```

- *cellTowerRSSI.txt*: almacenamos los cambios en la potencia recibida y el CID de la estación base a la que estamos conectados. Los parámetros siguen el orden: la fecha con foramto yyyymmdd hhmmSSsss, el CID y el nivel de potencia recibida en dBm.

```
1
2 20120429 120547312       245      -89.0
```

- *triangulationTime.txt*: guardamos el tiempo que tardan cada uno de los algoritmos de trilateración en calcular los centros de acuerdo con le siguiente orden: la fecha con foramto yyyymmdd hhmmSSsss, incentro, centroide, circuncentro y ortocentro.

```
1
2 20120429 120643673       518799   2624511  1007080
   4028320
```

- *gpsAndCellidCoordinates.txt*: para cada actualización en la posición proporcionada por Google o por un cambio de la celda a la que estamos conectados. Guardamos los parámetros en el siguiente orden: la fecha con foramto yyyymmdd hhmmSSsss, la latitud y la longitud de la posición del usuario según el GPS, la latitud y la longitud de la estación base a la que estamos conectados y de las dos estaciones vecinas de las que recibimos mayor nivel de potencia, y la latitud y la longitud de los cuatro centros calculados (incentro, centroide, circuncentro y ortocentro).

```
1
2 20120429 120643654       40486761         -3656978
            40486761         -3656978         40488139
```

```
            -3671697        40356408        -3907021
            40484276        -3670372        40443769
```

- *ttff.txt*: guardamos una marca de tiempo con foramto yyyymmdd hhmmSSsss y el tiempo que ha tardado el GPS en localizarnos por primera vez.

```

20120428 150051363      35202
```

- *wifiInfo.txt*: almacenamos información referente a la estación WiFi a la que estamos conectados siguiendo este orden: el identificador de la estación (Service Set IDentifier, SSID), el nivel de potencia recibida en dBm, el identificador básico de la estación WiFi (Basic Service Set IDentifier, BSSID), la dirección de red (Media Access Control address, MAC), la dirección IP, el identificador de la red inalámbrica, la velocidad del enlace y el identificador oculto de la estación WiFi.

```

7895651 WiFi-UC3M       -63     00:27:0d:56:03:10
          38:E7:D8:D9:19:CA       384721930       2
          54      false
```

- *wifiHeighboursInfo.txt*: guardamos información referente a las estaciones WiFi vecinas a la que estamos conectados con el siguiente orden: el SSID, el nivel de potencia recibida en dBm, el BSSID, las capacidades (información sobre la autenticación, la administración de la clave y los esquemas de cifrado) y la frecuencia del canal de comunicación.

```

7895651 GAST-WIFI       -51     00:18:f8:7a:31:83
          [WPA-PSK-TKIP+CCMP]     2462
```

Los ficheros *cellTowerInfo.txt*, *cellTowerNeighbours.txt*, *cellTowerRSSI.txt* y *triangulationTime.txt* sólo se crean cuando la opción de localización por GSM está activada, los ficheros *wifiInfo.txt* y *wifiHeighboursInfo.txt* sólo se crean si el usuario pulsa sobre la opción "Scan WiFi Access Points" del menú y el fichero *ttff.txt* cuando la localización por GPS está activada. Mientras que el fichero *gpsAndCellidCoordinates.txt* siempre se creará si cualquiera de los dos tipos de localización están activados.

3.4.6. Críticas y problemas encontrados

Durante el desarrollo de la aplicación Android se encontraron una serie de problemas a los que tuvimos que hacer frente.

- Escritura en ficheros. Nuestra aplicación escribe en ficheros la información de localización recogida. Para que el usuario pueda acceder a estos ficheros, éstos deben residir en la tarjeta de memoria del teléfono para posteriormente ser transferidos a un ordenador y procesarlos para el análisis de la aplicación. Esta escritura en la tarjeta sólo se realiza en el caso de que el teléfono móvil tenga tenga montada dicha tarjeta, esto quiere decir que si el usuario conecta el móvil a un ordenador y está leyendo o interactuando con la tarjeta de memoria del dispositivo móvil, esta se encuentra desmontada, y por tanto, nuestra aplicación no puede escribir en dicha tarjeta. Para evitar este problema, durante el ciclo de ejecución de la aplicación, la información se escribe en ficheros que residen en la memoria interna del propio teléfono, y al pulsar la opción "Salir" del menú de la aplicación se realiza un volcado de los ficheros a la memoria externa del teléfono, es decir, a la tarjeta de memoria.

- Peticiones HTTP. Las peticiones que se realizan contra la API de Google para obtener la latitud y la longitud de las estaciones base, no se realizan si el usuario está conectado a una red WiFi debido a que se tratan de una conexión HTTPS y el usuario debe estar autenticado en la red WiFi para que estas conexiones se establezcan. Por lo tanto, el dispositivo móvil debe estar dotado de conexión de datos para poder realizar estas peticiones con éxito, o bien, estar conectado a una red WiFi que permita realizar peticiones HTTPS. No se trata de algo crítico puesto que esta aplicación está pensada para que funcione tanto en interiores como en exteriores, y si el usuario posee conexión de datos podrá ser localizado en todo momento, sin embargo, si el usuario depende de tener una conexión WiFi para poder conectarse, en escenarios exteriores no podrá consegirlo la mayoría de las veces ya que las redes WiFi abiertas que existen actualmente no suelen permitir conexiones de tipo HTTPS.

- Eventos de cambio de celda GSM. En este caso, el problema residía que si el terminal bloqueaba la pantalla los eventos de cambio de celda no se obtenían, y por tanto, se dejaba de localizar al usuario. Como solución se activó la propiedad `android:keepScreenOn` de la actividad encargada de mostrar las localizaciones del usuario en pantalla para evitar que el terminal se bloqueara.

- Información sobre las estaciones base vecinas. Como se ha ido explicando a lo largo del proyecto, nuestra aplicación utiliza la información de las estaciones vecinas para poder calcular mediante métodos de trilateración la posición del usuario. Durante esta fase del proyecto se detectó que no todos los dispositivos Android sobre los que se instaló la aplicación nos porporcionaban información acerca de las estaciones vecinas. Esto se trata de un *bug* que actualimente está por resolver por el equipo de Google pero que sin embargo no tiene una solución rápida. Al tratarse de eventos procendentes de la API Android nativo del teléfono cada fabricante realiza su implementación personalizada de este API, y por tanto, dependemos de que, por ejemplo, Samsung implemente estos eventos en sus teléfonos para que nosotros podamos obtener información. Hasta ahora Google se ha pronunciado comunicando que todos los teléfonos fabricados por ellos tienen implementada esta parte de la API de Android, y por tanto, esto nos perjudica porque no hace tan portable nuestra aplicación. Para seguir el estado de este problema consultar la siguiente url `http://code.google.com/p/android/issues/detail?id=24136&q=getNeighboringCellInfo&colspec=ID%20Type%20Status%20Owner%20Summary%20Stars`

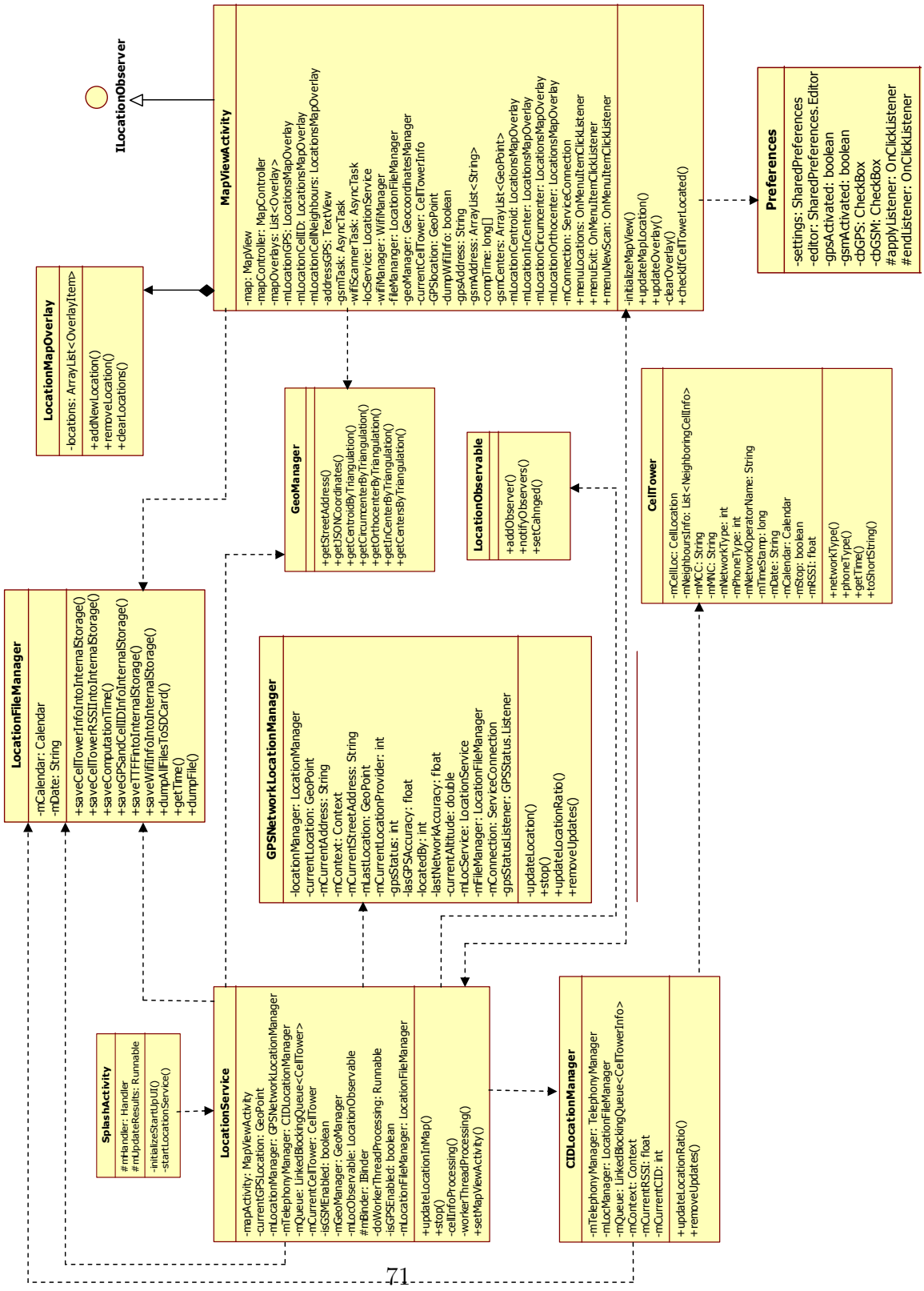


Figura 3.12: Diagrama UML de DroidTriangulation

Capítulo 4

Entorno de Pruebas

El objetivo de este capítulo es describir los escenarios en los que se han realizado medidas con la aplicación, tanto su ubicación como condiciones, qué características se han medido para realizar un posterior análisis y los elementos que se han utilizado para ello.

4.1. Dispositivos empleados

Como se ha ido mencionando a lo largo de este trabajo, la aplicación está orientada a dispositivos con sistema operativo Android, por lo tanto, se ha escogido un móvil HTC Desire con las siguientes características:

Velocidad de Proceso de la CPU - 1GHz
Plataforma - Android 2.2.2
Almacenamiento - ROM 512 MB
Almacenamiento - RAM 576 MB
Resolución - 480 x 800 WVGA
Banda de Red en Europa - HSPA/WCDMA y GSM
Internet - 3G (Velocidad descarga 7.2Mbps, Velocidad de subida 2Mbps)
Internet - GPRS (Velocidad de descarga 114Kbps)
Internet - EDGE (Velocidad de descarga 560Kbps)
Internet - WiFi IEEE 802.11 b/g

Tabla 4.1: Características HTC Desire

Tal y como se comenta en [10], el GPS integrado en los dispositivos móviles es menos preciso que un disposito GPS autónomo, de modo que, se ha empleado un TomTom One, puesto que nos permitía obtener nuestra posición

como par latitud-longitud, considerando la localización que nos proporcionaba como posición de referencia para medir la precisión de la aplicación. Las características de este dispositivo son las siguientes:

Velocidad de Proceso de la CPU - 380MHz
Pantalla táctil LCD TFT - 320 x 240
Dimensiones - 110mm x 89mm x 34 mm
Almacenamiento - RAM 32 MB
Receptor GPS integrado basado en SiRF Star III
Tarjeta de memoria (tarjeta SD) con software y mapas

Tabla 4.2: Características HTC Desire

Para cronometrar cada una de las mediciones, se empleó un cronómetro independiente y ajeno al que viene integrado en el dispositivo HTC Desire.

4.2. Escenarios

A la hora de realizar medidas y evaluar la precisión de los centros calculados con respecto a la posición obtenida por el dispositivo GPS autónomo, interesa llevar a cabo pruebas tanto en zonas urbanas donde las estaciones base se encuentran separadas en distancias del orden metros como en zonas rurales donde las estaciones base están separadas varios kilómetros entre ellas. A continuación, se muestran las localizaciones en las que se tomaron las medidas.

4.2.1. Localización urbana

Uno de los principales objetivos de la aplicación es evaluar el funcionamiento tanto en interiores como en exteriores, por ello, se han escogido diferentes lugares dentro del campus de la Universidad Carlos III. Las medidas en interiores se han realizado en el laboratorio 4.1.A01, cuya lozalización se muestra en la figura 4.1.

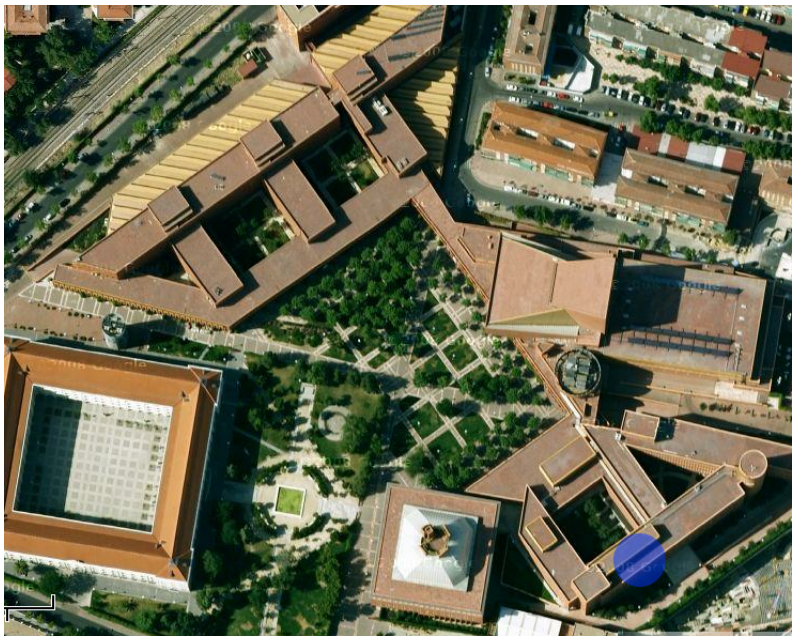


Figura 4.1: Situación del laboratorio 4.1.A01

Las medidas en exteriores se hicieron en tres puntos del campus de la Universidad Carlos III con el objetivo de ser localizados empleando estaciones base distintas tal y como se muestra en la figura 4.2.

Figura 4.2: Puntos en los que se realizaron medidas en el campus

4.2.2. Localización rural

Las pruebas se realizaron en un pueblo de la provincia de Ávila llamado Villafranca de la Sierra. En este escenario no se pudieron realizar medidas en interiores por la ausencia de cobertura en el interior de las viviendas. Sin embargo, se tomaron medidas exteriores en un sólo punto debido a que al tratarse de un pueblo de reducida extensión, en puntos opuestos del mismo se recibía señal siempre de la misma estación base, y además no fue posible recibir señal de celdas vecinas.

En la figura 4.3 se representa la distancia entre el punto donde se tomaron las medidas (A) y la situación de la estación base a la cual estábamos conectados (B), que como se puede observar, aproximadamente, se encuentra a 18.4 Km de distancia.

Figura 4.3: Distancia entre la estación base y el punto origen de las medidas

En la siguiente figura 4.4 se muestra un plano ampliado de la extensión del pueblo.

Figura 4.4: Plano de Villafranca de la Sierra

4.3. Medidas

Esta aplicación está orientada a medir una serie de caractrerísticas con el fin de realizar un análisis de las prestaciones frente a las aplicaciones de localización existentes en el mercado. Estas medidas se realizan para posiciones estáticas, puesto que el objetivo de la aplicación no es realizar un seguimiento de la localización del usuario.

Estas medidas se realizan en varios ciclos y en distintas localizaciones descritas anteriormente. El período de duración de cada ciclo es de 10 minutos, ya que se considera que en dicho plazo la aplicación ha calculado la posición del usuario varias veces para permitir realizar una ponderación en el análisis posterior de las prestaciones. Aunque se toma un minuto de margen al principio y al final de la medida para poder descartar posibles transitorios en dicho análisis.

A pesar de que en un principio se pretendía posicionar al usuario mediante estaciones base WiFi, finalmente se ha descartado por la no disponibilidad de medios suficientes para poder tomar las medidas. Simplemente se han realizado barridos de información de algunos sitios para comprobar la información que nos proporciona Google en estos escenarios.

A continuación, se presentan las características que se han medido y los medios que se han empleado para ello.

4.3.1. Precisión

Cuando hablamos de precisión, nos refererimos a cuán cercana es la localización que hemos obtenido respecto a la proporcionada por un dispositivo GPS autónomo midiendo la diferencia en metros. Esta precisión varía en función de la distancia del usuario a las tres estaciones base más cercanas, que en nuestro caso, son aquellas de las cuales el dispositivo móvil recibe mayor nivel de señal. Por ello, para medir esta característica se ha realizado en dos ámbitos: urbano y rural, puesto que la densidad de estaciones base por kilómetro cuadrado no es la misma en ciudades como en pueblos.

El objetivo es medir el error cometido por la aplicación en metros, por este motivo para cada actualización de posición, bien mediante GPS (o red) o mediante GSM, se guardan en fichero todas las posiciones para un instante concreto de tiempo:

- Posición obtenida por GPS (o red).
- Posición de la estación base a la que estamos conectados.
- Posición de las dos estaciones base vecinas con mayor nivel de señal obtenida.
- Cuatro posiciones obtenidas por métodos de trilateración.

4.3.2. Consumo de batería

Una de las motivaciones de este proyecto es ofrecer una aplicación cuyo consumo de batería sea menor que el de las aplicaciones de localización

existentes. Por ello, se ha utilizado la aplicación *PowerTutor* que puede ser descargada gratuitamente de *Google Play* y que nos permite obtener el consumo de batería de algunos componentes del móvil como son la CPU, la interfaz de red, la pantalla, el receptor de GPS y las diferentes aplicaciones. Para más información consultar `http://powertutor.org`.

PowerTutor es una aplicación para dispositivos Android que realiza un seguimiento del consumo aproximado de batería instantáneo de algunos componentes del teléfono y de las aplicaciones instaladas en él. Esta aplicación muestra gráficas de dicho consumo de algunos de los componentes del teléfono como son la CPU, la interfaz de red o la pantalla. Está compuesta por varias vistas, una de ellas muestra distintos tipos de gráficas con el consumo instantáneo tal y como se muestra en la figura 4.5.

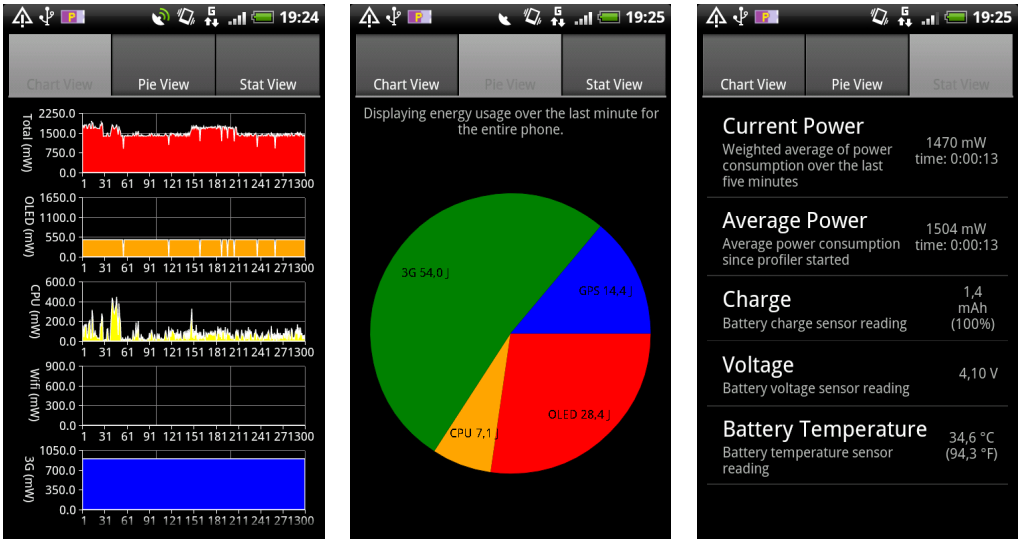


Figura 4.5: Patalla de consumo de batería del sistema

Además, permite conocer el consumo de batería de cada aplicación en función del componente del sistema que figuran en pentañas distintas como se puede observar en la figura 4.6.

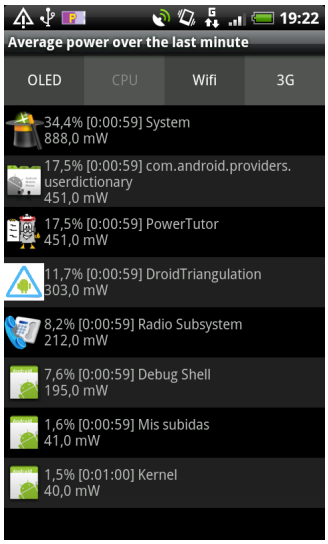


Figura 4.6: Patalla de consumo de batería de aplicaciones

En nuestro caso, lo que nos interesa es el archivo de texto que se puede obtener con datos de consumo para posteriormente analizar el consumo de batería de nuestra aplicación. A continuación se muestra un extracto de la estructura de este fichero.

```
begin 0
total-power 474
OLED 474
Wifi 0
3G 0
GPS 0
associate 10001 com.google.uid.shared:10001@8
associate 10002 com.htc.bg.uid.shared:10002@8
```

Como se puede observar la estructura no es muy amigable de cara a procesar estos datos en MATLlab, por lo tanto, se creó un programa UNIX para escoger los datos referentes a la marca de tiempo (*begin*), el consumo de la pantalla (*OLED*), de la wifi (*WiFi*), del GPS (*GPS*), de 3G (*3G*) y el dato de consumo total de potencia (*total-power*), y organizarlo por columnas.

Para obtener los datos de consumo de nuestra aplicación se realizó una primera medida en la que el teléfono se encontrase en las mismas condiciones que en las que se encontraría durante las medidas pero sin estar ejecutándose nuestra aplicación:

- Batería del teléfono completamente cargada.

- La pantalla del teléfono encendida, con el nivel de brillo al máximo y sin bloquear puesto que se considera que el usuario durante el uso de la aplicación estará mirando a la pantalla continuamente.
- Con la WiFi, el GPS y los datos móviles desactivados.
- Con la tarjeta SIM.
- Ninguna aplicación ejecutándose, excepto con la aplicación de medida del consumo de batería.

Esta medida nos sirve de referencia de consumo del teléfono cuando nuestra aplicación no está ejecutándose. Durante el ciclo de toma de medidas esta aplicación estará monitorizando el consumo instántaneo de batería, y al restar los datos obtenidos durante las medidas con dicha referencia, podremos obtener un consumo aproximado de nuestra aplicación.

Por otra parte, a la hora de realizar estas medidas, primero debíamos iniciar la aplicación *PowerTutor* y esperar un minuto hasta que se estabilizara, y tras finalizar las medidas, debíamos cerrar la aplicación y esperar otro minuto para que volviese a estabilizarse.

Puesto que la aplicación permite posicionar mediante el uso del GPS del dispositivo, mediante trilateración y ambas opciones simultáneamente, se tomaron medidas independientes para cada una de estas casuísticas para facilitar el análisis del consumo de batería.

4.3.3. Latencia

En este caso, el término latencia hace referencia al tiempo que tarda la aplicación en obtener una primera localización del usuario por primera vez. En nuestro caso, nos interesa medir este tiempo cuando estamos siendo localizados mediante GPS y cuando estamos siendo localizados mediante métodos de trilateración.

En el caso del GPS, cuando la aplicación se inicia por primera vez tarda varios segundos en obtener una posición, y una vez obtiene una localización para el usuario, se quedará almacenado en memoria la última posición que se obtuvo en la última ejecución de la aplicación. Por lo tanto, para la realización de esta medida, había que borrar los datos de la aplicación para evitar que la aplicación utilizase la última posición obtenida para el usuario en su última ejecución, puesto que esto supone una localización inmediata del usuario.

Sin embargo, esto no ocurre en el caso del posicionamiento mediante métodos de trilateración ya que la API de localización de Android no almacena en memoria la información referente a la última estación base a la que se

conectó el usuario. Y por lo tanto, al iniciar la aplicación debemos esperar a obtener la información sobre la estación base a la que estamos conectados, y si se nos proporciona información sobre dos estaciones vecinas, la aplicación procede a pedir la latitud y la longitud de cada una de las estaciones para realizar los cálculos de los centros. Hay que tener en cuenta que este procedimiento se ve afectado por varios factores:

- Cobertura. Afecta a la hora de realizar las peticiones HTTP contra la API de localización de Google, cuanto más baja sea la cobertura, más tardará la comunicación HTTP en establecerse.

- Número de estaciones vecinas. No siempre obtendremos información de al menos dos estaciones vecinas, de tal modo que, si esto ocurre al iniciar la aplicación, provoca que el usuario no pueda ser localizado y se muestra la posición de la estación a la que estamos conectados como medida preventiva.

En el caso de la localización mediante GPS (o red), Android nos proporciona un mecanismo de escucha de eventos que tras obtener la primera vez la posición del usuario podemos recuperar el tiempo que ha tardado. Mientras que para determinar esta medida en el caso del posicionamiento mediante métodos de trilateración debemos fijarnos en la marca de tiempo de los ficheros recogidos por la aplicación para calcular el tiempo que conlleva obtener la información de la estación base a la que estamos conectados, hacer tres peticiones HTTP para obtener la longitud y la latitud de las tres estaciones base implicadas, y el tiempo de cómputo de los cuatro centros que se calculan mediante algoritmos de trilateración.

En la siguiente tabla se resumen el número de medidas que se han realizado para valorar esta característica.

Escenario	Situación	GPS	GSM	Duración(min)	Ejecuciones
Urbano	Interior	x	x	10	2
Rural	Interior	x	x	10	2
Urbano	Exterior	x	x	10	2
Rural	Exterior	x	x	10	2

Tabla 4.3: Resumen de medidas realizadas

4.3.4. Tiempo de cómputo de los algoritmos de trilateración

Otra de las principales características de este proyecto es que a diferencia de las aplicaciones existentes basadas en localización que hacen uso tanto de GPS como de las coordenadas obtenidas usando el CID de la estación base más cercana, nuestra aplicación añade un tiempo adicional de computación de los cuatro centros calculados para la posición del usuario: centroide, incentro, circuncentro y ortocentro.

De modo que, la aplicación escribe en un fichero el tiempo, en milisegundos, que tarda en calcular dichos centros para posteriormente analizar el impacto que tiene este cálculo sobre las prestaciones de la aplicación y valorar su mejora de cara al futuro.

4.3.5. Resumen

A continuación se presenta la tabla 4.4 que resume brevemente el conjunto de medidas realizadas para realizar un posterior análisis de las prestaciones de los algoritmos de trilateración

4.3.6. Críticas y problemas encontrados

En la fase de pruebas del proyecto nos encontramos con el principal problema de que el dispositivo GPS integrado que empleamos para obtener la posición real del usuario no nos proporcionaba localización en interiores. Como solución optamos por situar el cursor sobre un mapa de Google y señalar donde nos encontrábamos para obtener las coordenadas de dicho punto, y consideramos dicha posición como posición real del usuario en interiores.

Escenario	Situación	Localización	GPS	GSM	WiFi	Duración (min)	Ejecu.
Urbano	Interior	Despacho 4.1.A01	x			10	2
Urbano	Interior	Despacho 4.1.A01		x		10	4
Urbano	Interior	Despacho 4.1.A01	x	x		10	9
Urbano	Interior	Despacho 4.1.A01			x	10	2
Urbano	Exterior	Jardines Edificio Sabatini	x			10	2
Urbano	Exterior	Jardines Edificio Sabatini		x		10	2
Urbano	Exterior	Jardines Edificio Sabatini	x	x		10	2
Urbano	Exterior	Edificio Juan Benett	x	x		10	2
Urbano	Exterior	Edificio Gimnasio	x	x		10	2
Urbano	Exterior	Edificio Biblioteca	x	x		10	2
Rural	Interior	Villafranca de la Sierra	x			10	2
Rural	Interior	Villafranca de la Sierra		x		10	2
Rural	Interior	Villafranca de la Sierra	x	x		10	2
Rural	Interior	Villafranca de la Sierra			x	10	2
Rural	Exterior	Villafranca de la Sierra	x			10	2
Rural	Exterior	Villafranca de la Sierra		x		10	2
Rural	Exterior	Villafranca de la Sierra	x	x		10	2

Tabla 4.4: Resumen de medidas realizadas

Capítulo 5

Análisis de resultados

El objetivo de este capítulo es realizar un análisis de la viabilidad de posicionar al usuario utilizando algoritmos de trilateración en lugar de GPS, mediante el estudio de gráficas relacionadas con las carácterísticas que se quieren estudiar y que se han ido nombrando a lo largo de esta memoria.

5.1. Análisis de la precisión

Uno de los objetivos de este proyecto es comparar el margen de error que se pueda cometer empleando métodos de trilateración para estimar la posición de un usuario frente a la posición proporcionada por el dispositivo GPS del móvil o la calculada por Google con información de redes inalámbricas.

En la siguiente figura 5.1 se muestra un mapa de las posiciones que se han calculado durante un ciclo de medida.

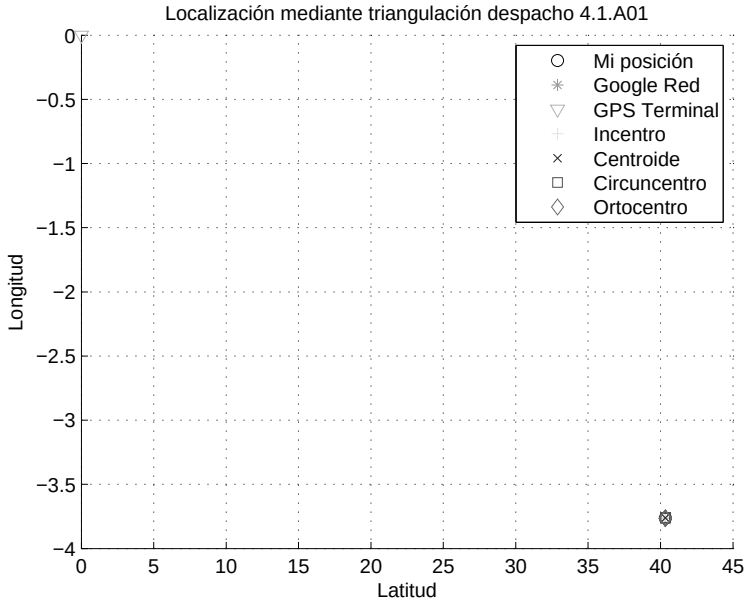


Figura 5.1: Mapa de posiciones en el despacho 4.1.A01

El punto considerado como "Mi posición" es la localización obtenida a través de la página web de *Google Maps* que nos permite conocer la latitud y longitud de un punto que escojamos del mapa. En este caso, el dispositivo GPS autónomo no fue capaz de localizarnos al estar situados en el interior de un edificio, por ello, optamos por seleccionar dicho punto como posición fiable. Por este mismo motivo el GPS integrado en el teléfono móvil tampoco fue capaz de lozalizarnos, tal y como se puede observar en la figura 5.1 donde dicha localización aparece en la esquina superior izquierda correspondiente a latitud cero y longitud cero.

En la siguiente figura 5.2 se muestra una vista ampliada de las localizaciones obtenidas en el despacho 4.1.A01 sin incluir la posición del GPS integrado del terminal.

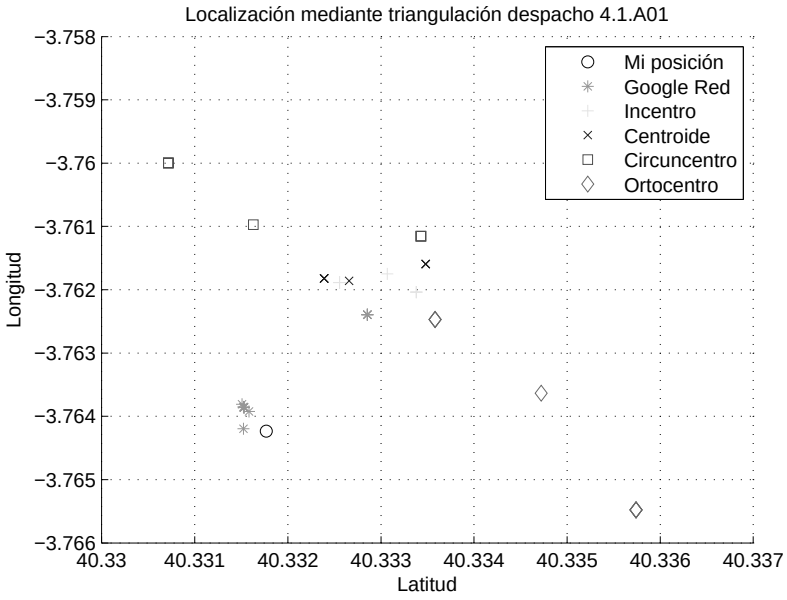


Figura 5.2: Mapa ampliado de posiciones en el despacho 4.1.A01

Podemos observar que aunque nos encontremos dentro de un edificio las posiciones calculadas por Google están más cercanas de la posición real que las calculadas por métodos de trilateración. Cabe destacar que las localizaciones proporcionadas por Google fueron calculadas mediante red en lugar del dispositivo GPS.

Las posiciones más alejadas corresponden al ortocentro y al circuncentro, esto se debe a la forma del triángulo que componen las tres estaciones base que se están utilizando en ese momento para los cálculos, ya que se encuentran formando un triángulo obtuso, y por tanto, estos centros se sitúan fuera del área comprendida por dicho triángulo, tal y como se puede apreciar en la siguiente figura 5.3:

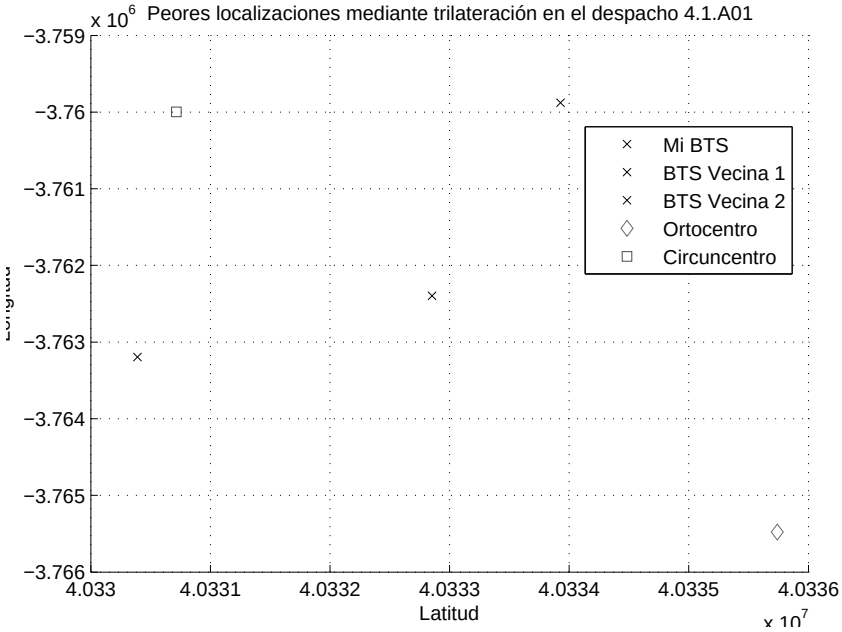


Figura 5.3: Mapa de las peores posiciones en el despacho 4.1.A01

Para calcular el error básandonos en la distancia entre las posiciones obtenidas por el GPS integrado, Google y trilateración a la posición del usuario, hemos utilizado lo que se conoce como ortodrómica (GCD, Great-Circle Distance). La diferencia entre este tipo de distancia y la distancia euclídea es que es el camino más corto entre dos puntos de la superficie terrestre, que es lo que realmente estamos considerando al representar las localizaciones mediante su latitud y longitud. La siguiente gráfica 5.4 representa el error medio cometido para cada una de las posiciones y su desviación típica:

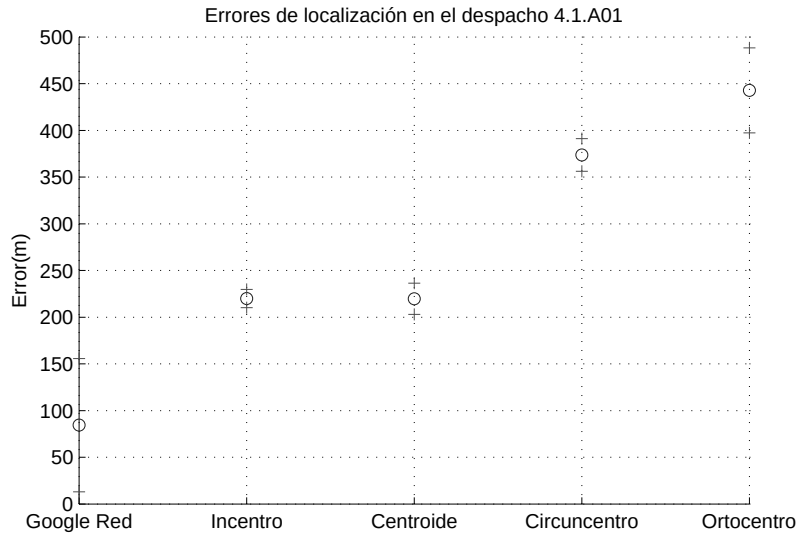


Figura 5.4: Error cometido en las posiciones en el despacho 4.1.A01

En este caso, el error máximo cometido mediante posicionamiento a través de la red por Google es de 197.1158m, mientras que el error máximo cometido mediante métodos de trilateración es el correspondiente al ortocentro y es de 454.1477m. Por otra parte, el error mínimo cometido mediante localización por red es de 27.3322m y mediante trilateración es de 215.8338m correspondiente al centroide. En la siguiente tabla 5.1 se resumen los errores cometidos:

Método	Error Máximo (m)	Error Mínimo (m)
Google Red	197.1158	27.3322
Incentro	258.3982	217.4610
Centroide	293.9235	215.8338
Ortocentro	454.1477	250.9559
Circuncentro	377.9174	277.1045

Tabla 5.1: Errores cometidos en interiores según GCD

Por otra parte se puede observar que el centro con menor desviación típica es el incentro. Esto implica que este tipo de centro es menos sensible a la forma del triángulo formado por las tres estaciones base. Sin embargo, el ortocentro confirma su dependencia de dicha forma al tener una desviación típica mayor.

En el caso de un escenario exterior, el mapa de posiciones obtenido es el de la siguiente figura 5.5:

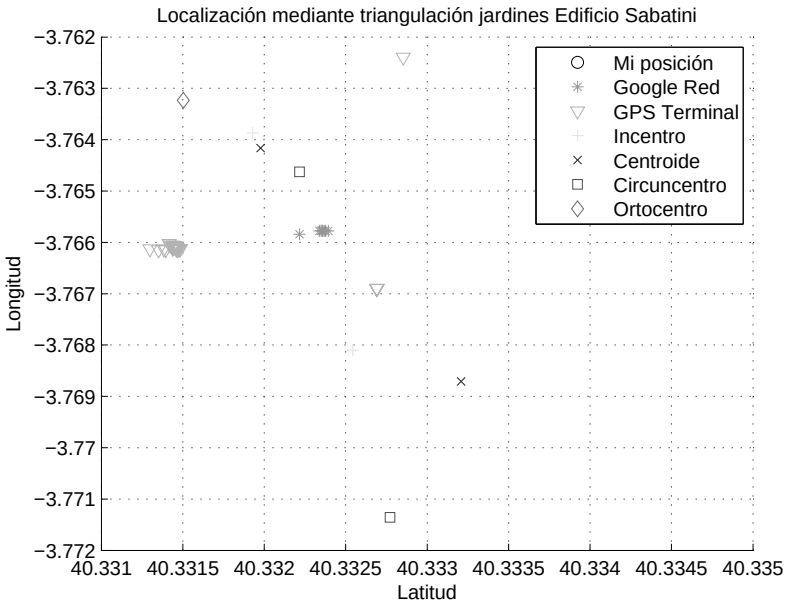


Figura 5.5: Mapa de posiciones en los jardines del edificio Sabatini

A diferencia del caso anterior, ahora la localización considerada como "Mi posición" es la que hemos obtenido con el dispositivo GPS autónomo, y como era de esperar la posición proporcionada por el GPS integrado en el terminal es aproximadamente la misma la mayoría de las veces, mientras que la localización realizada por red es menos preciso. En la siguiente gráfica 5.6 se representa el error medio cometido por cada una de las posiciones y su desviación típica.

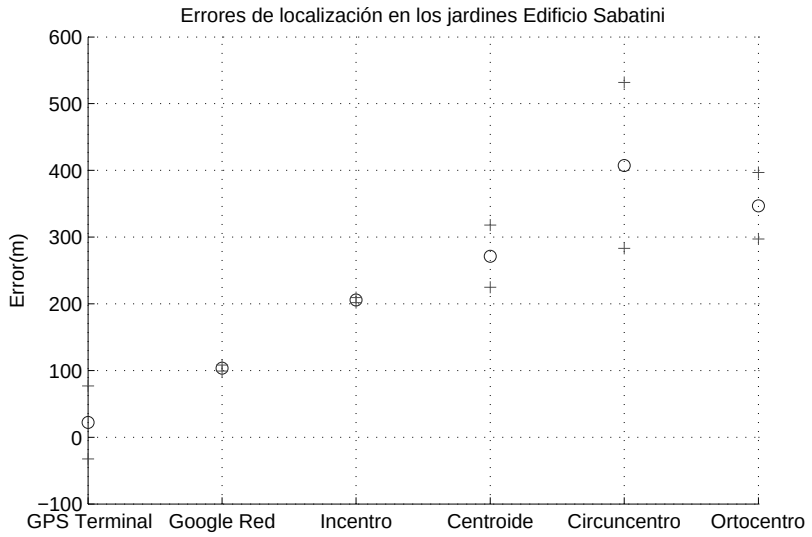


Figura 5.6: Error cometido en las posiciones en Jardines Edificio Sabatini

Cabe destacar que una de las posiciones calculadas mediante métodos de trialateración con la que se ha cometido un error alto es con el centroide en uno de los casos. Esto se debe a que el triángulo formado por las estaciones base, o bien, éstas se encuentran muy separadas entre ellas, o bien, la posición que estamos considerando como fiable se encuentra en una de las puntas del triángulo.

En este caso, los errores son ligeramente menores tal y como se puede observar en la siguiente tabla 5.2 puesto que los errores máximos vienen marcados por lo que se considerarían *outlayers*.

Método	Error Máximo (m)	Error Mínimo (m)
GPS Terminal	351.7606	0.5812
Google Red	108.3165	87.9010
Incentro	207.3662	197.9815
Centroide	293.8612	175.7405
Ortocentro	371.1576	244.5239
Circuncentro	467.5589	152.1588

Tabla 5.2: Errores cometidos en exteriores según GCD

El error máximo cometido en esta medida es 467.5589m y se corresponde con el circuncentro y el mínimo es 0.5812m correspondiente a una posición

mediante el GPS integrado en el dispositivo móvil. Al igual que en el caso de la localización en interiores el incentro tiene la desviación típica menor que el resto, mientras que en este caso el centro con mayor desviación típica es el circuncentro y esto se debe a que el triángulo cuanto mayor sea su ángulo obtuso, mayor será la circunferencia que lo engloba, y por tanto, el error a cometer será mayor.

Como era de esperar el GPS en exteriores es más preciso que en interiores. Además, tanto en interiores como en exteriores sigue siendo más preciso que las posiciones calculadas mediante algoritmos de trilateración en algunos casos. Esto se debe a que estas posiciones dependen bastante del tipo de triángulo que se esté considerando en cálculo y de la distancia entre las tres estaciones base implicadas.

Dentro de los centros calculados mediante trilateración podríamos decir que el centroide es el que mejor ha funcionado en ambos escenarios. Sin embargo, esto no es completamente determinante puesto que el cáculo de estos centros depende mucho de la forma del triángulo compuesto por las tres estaciones base y la distancia entre ellas. Además, como se ha demostrado, tanto el otrocentro como el circuncentro son muy dependientes de la forma del triángulo puesto que en función de si se trata de un triángulo obtuso o no, nos situarán fuera de él o en su interior respectivamente.

A continuación, vamos a realizar un análisis de la precisión en entornos rurales. El principal problema que se encontró es que en interiores el GPS del dispositivo móvil no era capaz de obtener una posición para el usuario y además, no disponíamos de cobertura móvil, y por tanto, tampoco se podían aplicar métodos de trilateración.

Por ello, el análisis se va a centrar en analizar el comportamiento de la aplicación en exteriores. En la siguiente figura 5.7 se muestra un mapa con las localizaciones calculadas para una ejecución puesto que la posición del usuario no variaba y siempre era la misma como se comentará más adelante durante el análisis del error cometido.

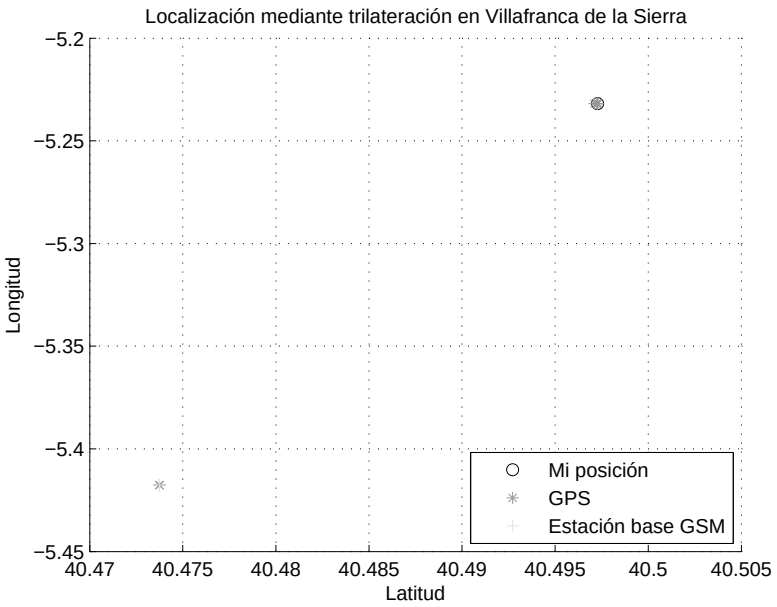


Figura 5.7: Mapa de posiciones en Villfranca de la Sierra

Como se puede observar, para el caso de la localización mediante GSM no se pudieron obtener los cuatro centros mediante algoritmos de trilateración debido a que la estación a la que estábamos conectados, al estar separada de las estaciones vecinas varios kilómetros, no obtenía el nivel suficiente de señal para que el dispositivo móvil recibiese información de dichas estaciones. Por lo tanto, en este caso la aplicación considera como localización del usuario la estación base a la que estamos conectados, y por ello, se puede ver que son varios kilómetros respecto a la posición real.

La siguiente figura 5.8 muestra el error cometido en cada una de las posiciones.

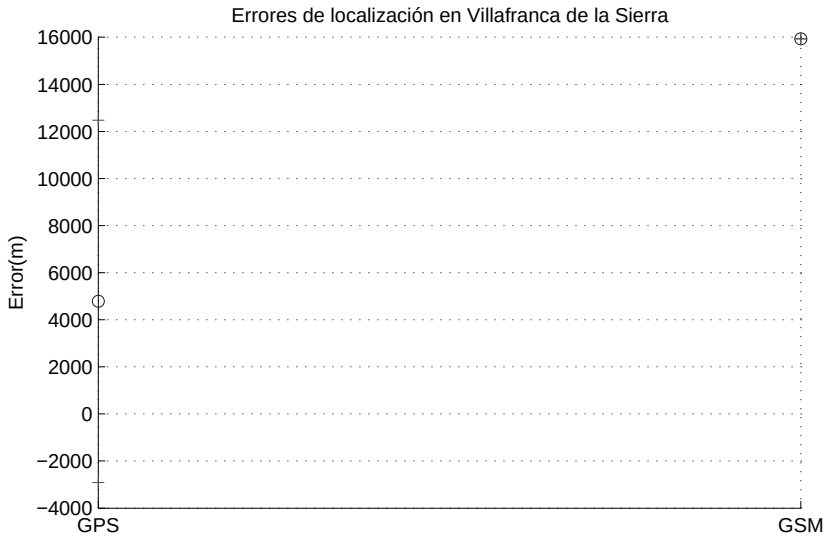


Figura 5.8: Mapa de posiciones en Villfranca de la Sierra

Como cabe esperar, la desviación típica en la localización por GSM es cero en todos los casos porque siempre nos ha situado en la localización correspondiente a la estación base a la que estamos conectados.

En la siguiente tabla 5.3 se muestra un resumen de los errores cometidos según GCD:

Método	Error Máximo (m)	Error Mínimo (m)
GPS	15937	1.7792
GSM	15937	15937

Tabla 5.3: Errores cometidos en Villafranca de la Sierra según GCD

Por lo tanto, como se puede observar el error mínimo cometido es de 1.77m correspondiente al GPS mientras que el error máximo cometido es el mismo tanto para GPS como para GSM, lo que da a enteder que el GPS del dispositivo móvil cuando no puede obtener posición para el usuario utiliza la localización por red y considera dicha posición la correspondiente a la estación base a la que está conectado, exactamente igual que nuestra aplicación.

5.2. Análisis del consumo de potencia

En el caso de un terminal móvil, el consumo de potencia está directamente relacionado con el consumo de batería. Como se ha comentado en la sección 4.3.2, el principal inconveniente de las aplicaciones basadas en localización es que el uso del dispositivo GPS consume mucha potencia reduciendo considerablemente la duración del ciclo de vida de la batería del terminal.

Inicialmente vamos a realizar un análisis comparativo del consumo de potencia que realiza el móvil en diferentes situaciones, tal y como se refleja en la gráfica 5.9.

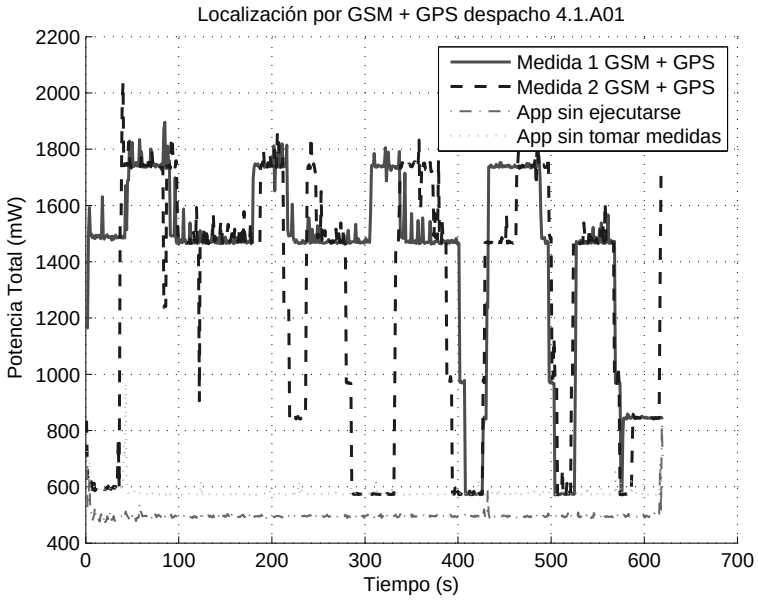


Figura 5.9: Localización GSM + GPS en el despacho 4.1.A01

Se puede observar que el consumo de batería del teléfono sin que se esté ejecutando nuestra aplicación es de aproximadamente 500mW con el móvil con la pantalla encendida y brillo al máximo. Cuando abrimos la aplicación pero desactivamos ambas opciones de localización, GPS y GSM, en las preferencias de la aplicación, se distingue una subida en el nivel de consumo de la aplicación a unos 600mW cuyo motivo es el que se muestre una vista de *Google Maps* en la pantalla y ésta se mantenga con el brillo a nivel máximo. Una vez que ejecutamos la aplicación, el nivel medio de consumo de la aplicación se ve incrementado a 1500mW ya que la aplicación se encuentra escribiendo en fichero información, actualizando las posiciones del usuario en la interfaz gráfica, calculado los centros a través de los métodos de trilateración, reci-

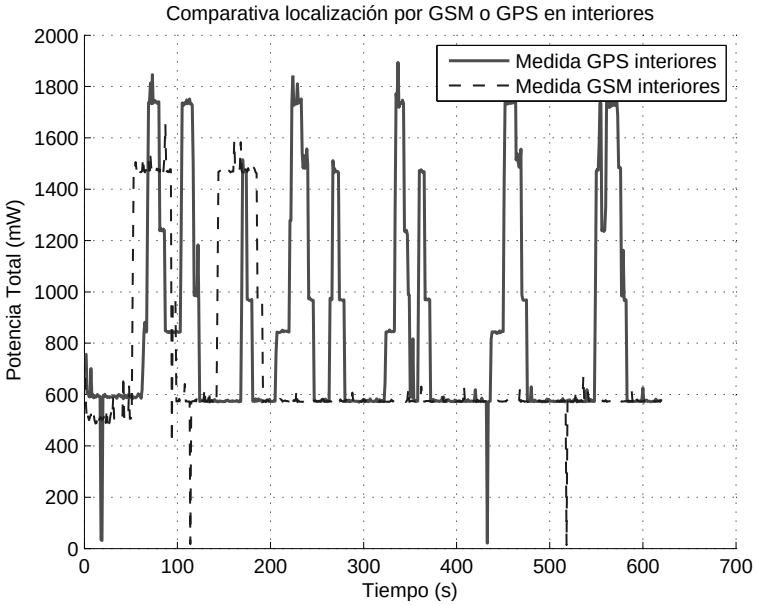


Figura 5.10: Comparativa de consumo de potencia en interiores

biendo actualizaciones de la posición del GPS y del cambio de celda, y por último, haciendo peticiones HTTP a Google para obtener las coordenadas de las tres estaciones base empleadas para los cálculos. Los niveles más altos de consumo se corresponden con la actualización de las localizaciones en la pantalla por motivo de las peticiones HTTP que se realizan y peticiones GPS, y los más bajos, se deben a que al estar realizadas las medidas en un interior, en esos momentos no se obtenía posición GPS y el GPS se desactiva porque no tiene señal de suficientes satélites.

Cabe destacar, que lo que más consume en la aplicación es la realización de peticiones HTTP y GPS, en lugar del propio pintado de éstas en la interfaz gráfica, lo que demuestra que aunque exista un único hilo de pintado, Android es capaz de optimizar su gestión para que consuma lo menos posible.

A continuación, vamos a centrar nuestro análisis en las diferencias de consumo de potencia en interiores y en exteriores. Para el caso de interiores, en la siguiente gráfica 5.10 destaca que el nivel de potencia consumida mediante la localización GSM es inferior a la cosumida utilizando solo la tecnología GPS. Esto refuerza nuestra teoría de que a pesar que para localizar al usuario mediante GSM la aplicación realiza peticiones HTTP contra el API de Google, ejecuta cuatro algoritmos de trilateración para calcular los centros y pinta un total de siete localizaciones en la interfaz gráfica (frente a una localización de GPS), el consumo es inferior.

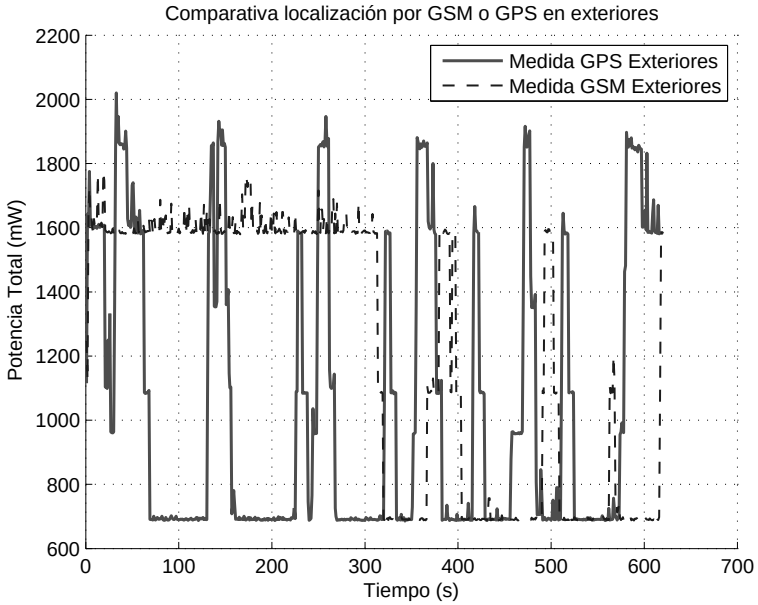


Figura 5.11: Comparativa de consumo de potencia en exteriores

Cabe destacar que sólo se visualizan dos picos correspondientes al posicionamiento mediante GSM. Esto se debe a que sólo de obtienen dos actualizaciones de cambio de celda durante la ejecución, y por tanto, no se vuelven a realizar peticiones HTTP contra la API de Google.

Si prestamos atención al caso de escenarios exteriores, se puede observar en la figura 5.11 que se cumple lo que comentabámos para el caso anterior. Por lo tanto, podemos concluir que la localización mediante algoritmos de trilateración consume menos potencia que la localización mediante GPS.

Destaca que durante los primeros 300 segundos de la ejecución el consumo de GSM sea tan alto. El motivo es que se reciben varias actualizaciones de cambio de celda seguidas, que se van almacenando en una cola tipo FIFO, y por tanto, cada una de estas actualizaciones conlleva petiticiones HTTP, por ello, el consumo se eleva.

Esta gráfica 5.12 muestra un resumen de las dos anteriores y nos permite conocer el valor medio consumido para cada uno de los casos.

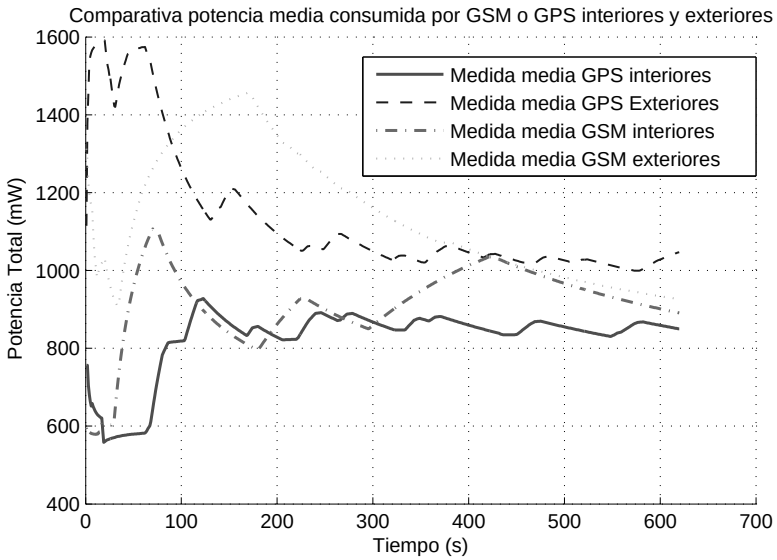


Figura 5.12: Comparativa de consumo de potencia en exteriores

Como se puede observar el consumo en interiores es menor que en exteriores, tanto para el caso de localización mediante GSM como mediante GPS. Además, el consumo de GSM en interiores en media es superior al de GPS, esto se debe a que el GPS a veces no obtiene posición para el usuario, mientras que mediante GSM siempre se obtiene una posición. En esta tabla 5.4 se muestran los valores medios de potencia consumida.

Método	Localización	Potencia media (mW)
Google Red	interiores	849.77
GPS	exteriores	1047.1
GSM	interiores	890.80
GSM	exteriores	926.177

Tabla 5.4: Potencia media consumida en interiores y exteriores

Adicionalmente, se puede observar que los pulsos correspondientes a una localización mediante GSM son de anchura variable frente a los pulsos correspondientes a una localización GPS. El principal motivo de esta diferencia es que GSM se ve afectada por el problema de la variación de la cobertura, esto quiere decir, que si apenas el móvil tiene cobertura las petitiones HTTP se verán ralentizadas, y por ello, en algunos casos tarda más en localizarnos.

En la siguiente gráfica 5.13 se muestra una comparación entre la potencia consumida en interiores y en exteriores cuando la aplicación tiene activado ambos tipos de localización, GPS y mediante algoritmos de trilateración.

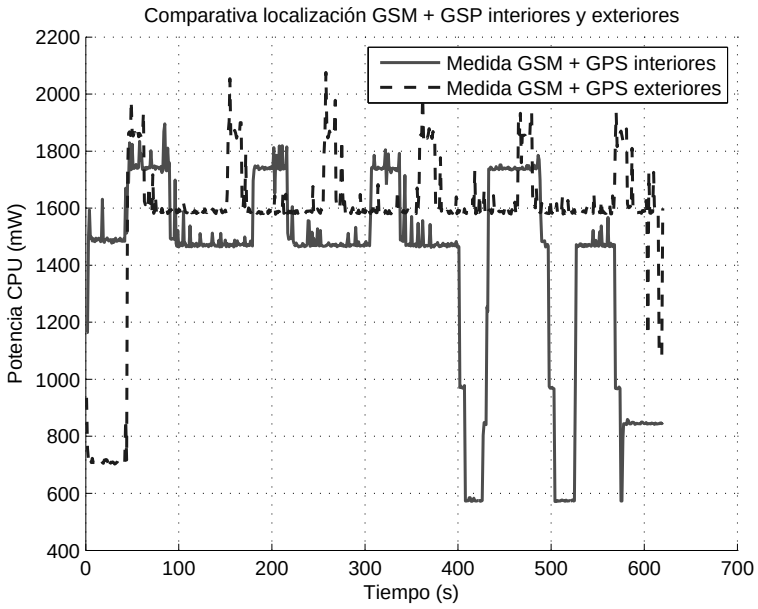


Figura 5.13: Localización GSM + GPS en interiores y exteriores

Se puede observar que la potencia consumida en interiores tiene un nivel medio más bajo que en exteriores, esto se justifica debido a que en interiores la comunicación mediante GPS no es siempre posible, y por tanto, nos está consumiendo menos batería que en el caso de exteriores ya que la aplicación está recibiendo posiciones GPS cada dos minutos. En lo respectivo a los niveles altos de ambas señales, destaca que en interiores la anchura de estos pulsos es mayor. El motivo de esta característica se corresponde con que en interiores la aplicación tarda más en localizarnos debido a que la cobertura es menor, y por tanto, las peticiones HTTP que se lanzan contra el API de Google para que nos proporcione las coordenadas de las estaciones base son más lentas, es decir, su velocidad de subida y bajada se ve ralentizada, y esto afecta directamente a que el tiempo que tarda la aplicación en calcular los cuatro centros es mayor.

Por otra parte, se ven varias bajadas en el nivel de potencia de la señal de interiores que se corresponden con el hecho de que la aplicación no nos haya podido localizar por GPS, y por tanto, el consumo de batería se ve reducido al no mantener una conexión tipo GPS.

5.3. Análisis de tiempo de cómputo de los algoritmos de trilateración

En este apartado vamos a realizar un análisis del tiempo de cómputo medio de cada uno de los centros calculados mediante métodos de trilateración principalmente en dos casos, sólo activada la opción de localización mediante GSM y activadas ambas opciones de localización (GPS y GSM), tanto en escenarios interiores como exteriores.

En el caso concreto de escenarios interiores y con sólo activada la opción de localización mediante GSM, los tiempos medios de computación son los que se muestran en la figura 5.14 y en la tabla 5.5:

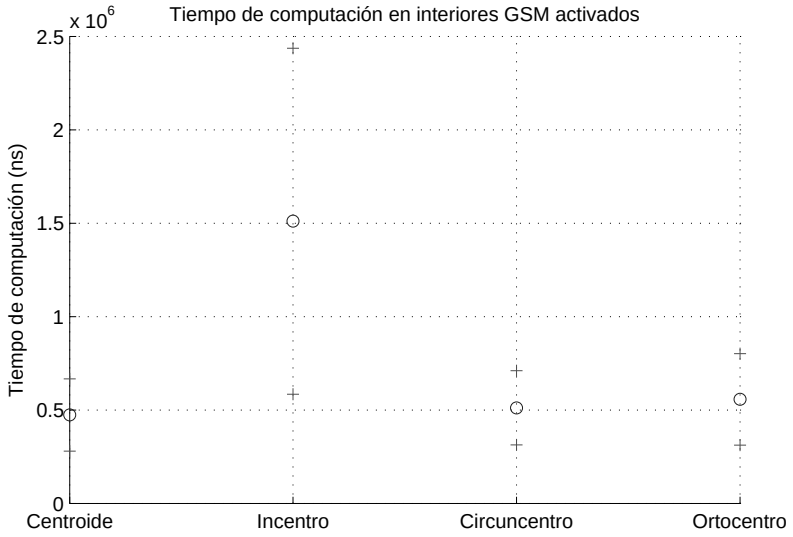


Figura 5.14: Tiempo de computación con GSM activado en interiores

Centro	Tiempo de computación (ms)
Incentro	1.510
Centroide	0.473
Circuncentro	0.511
Ortocentro	0.556

Tabla 5.5: Tiempo de cómputo con solo GSM en interiores

Cabe destacar que el tiempo de computación del incentro es casi tres

veces superior al resto. Esto se debe a que al aplicar la fórmula en código `Java` no disponíamos de la suficiente precisión para hacerlo en un solo paso, en este caso se corresponde con variables de tipo `long`. Como solución a este problema se optó por descomponer en pequeños pasos el cálculo para así no desbordar las variables, y por ello, es comprensible que tarde más en ejecutarse.

En la siguiente gráfica 5.15 y tabla 5.6 se muestra el tiempo de cómputo para el caso de interiores con la localización GPS y GSM activada:

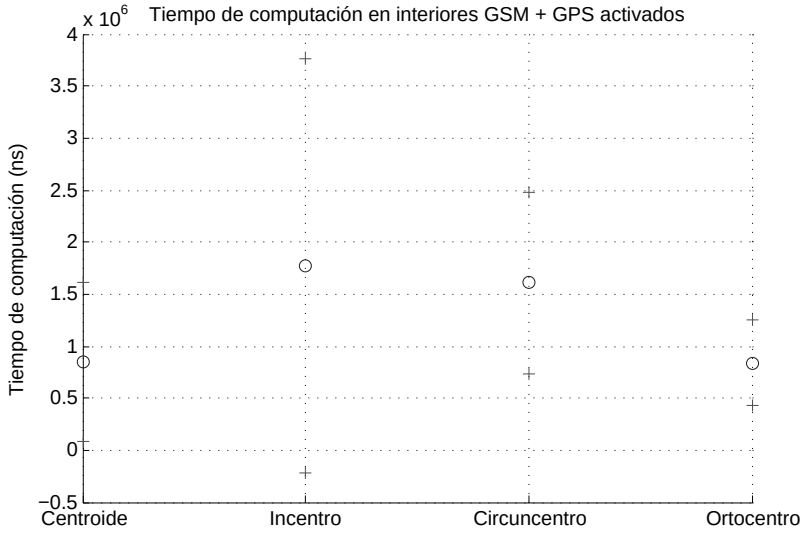


Figura 5.15: Tiempo de computación con GSM activado en interiores

Centro	Tiempo de computación (ms)
Incentro	1.770
Centroide	0.849
Circuncentro	1.606
Ortocentro	0.839

Tabla 5.6: Tiempo de cómputo con GSM + GPS activados en interiores

En este caso, el tiempo de cómputo se ha visto incrementado por el hecho de tener activado también el GPS que introduce una mayor carga en la ejecución de la aplicación, y que por lo que se puede observar, penaliza el cálculo de los centros.

En el caso de escenarios exteriores con sólo la opción de localización mediante GSM, los tiempos de cómputo se ven incrementados con respecto a escenarios interiores. Esto se debe a que el móvil cambia de estación base más a menudo en exteriores que en interiores, por tanto, se incremente la frecuencia de actualización de la posición del usuario y afecta a la capacidad de cómputo de la aplicación. Esto se ve reflejado en la siguiente gráfica 5.16 y en la tabla 5.7:

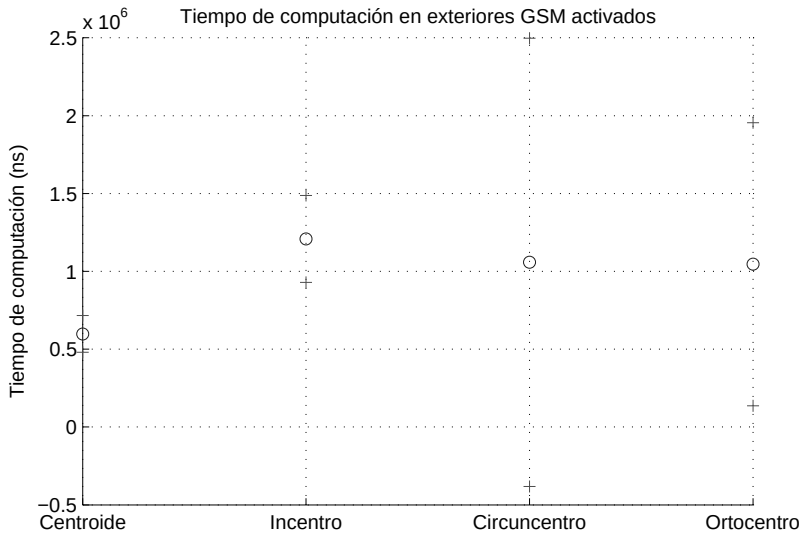


Figura 5.16: Tiempo de computación con GSM activado en exteriores

Centro	Tiempo de computación (ms)
Incentro	1.208
Centroide	0.598
Circuncentro	1.057
Ortocentro	1.045

Tabla 5.7: Tiempo de cómputo con GSM activado en interiores

Por último, en el caso de escenarios exteriores con ambas opciones activadas, se mantiene el hecho de que los tiempos son más elevados que en el caso de tener solo activado GSM como se puede observar en la gráfica 5.17 y en la tabla 5.8.

Cabe destacar que en el caso con ambas opciones activadas, el tiempo de cómputo en interiores es mayor que en exteriores porque el GPS en teriores

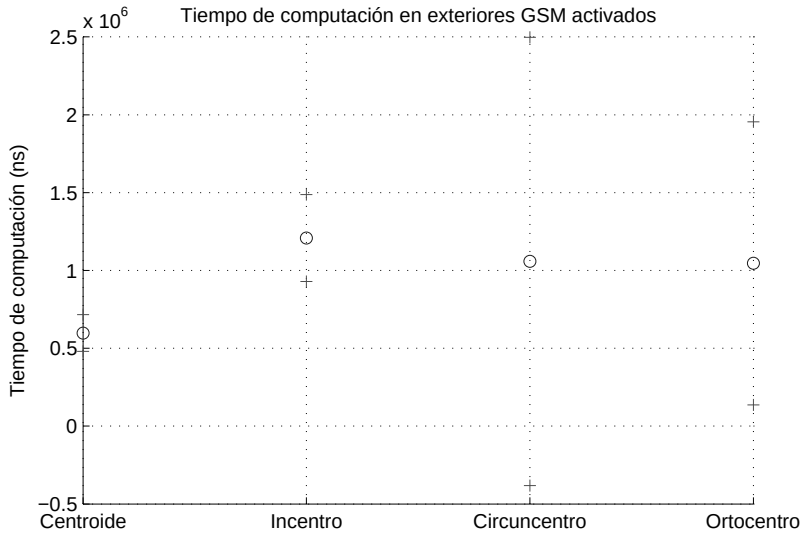


Figura 5.17: Tiempo de computación con GSM + GPS activados en exteriores

Centro	Tiempo de computación (ms)
Incentro	9.0210
Centroide	0.5875
Circuncentro	0.706
Ortocentro	0.611

Tabla 5.8: Tiempo de cómputo con GSM + GPS activados en exteriores

tarda más en sincronizar y termina haciendo uso de la red, mientras que en exteriores sí que el dispositivo GPS nos lacaliza de forma inmediata.

Finalmente, en la siguiente tabla 5.9 se muestra un resumen de los tiempos de computación de la aplicación.

Centro	Opciones	Localización	Tiempo de computación (ms)
Incentro	GSM	Interiores	1.510
Incentro	GSM + GPS	Interiores	1.770
Incentro	GSM	Exteriores	1.208
Incentro	GSM + GPS	Exteriores	9.0210
Centroide	GSM	Interiores	0.473
Centroide	GSM + GPS	Interiores	0.849
Centroide	GSM	Exteriores	0.598
Centroide	GSM + GPS	Exteriores	.5875
Circuncentro	GSM	Interiores	0.511
Circuncentro	GSM + GPS	Interiores	1.606
Circuncentro	GSM	Exteriores	1.057
Circuncentro	GSM + GPS	Exteriores	0.706
Ortocentro	GSM	Interiores	0.556
Ortocentro	GSM + GPS	Interiores	0.839
Ortocentro	GSM	Exteriores	1.045
Ortocentro	GSM + GPS	Exteriores	0.611

Tabla 5.9: Resumen de los tiempos de cómputo

5.4. Análisis de la latencia

Como se comentó en el capítulo 4 nuestra aplicación también tiene como cometido medir el TTFF tanto para la localización mediante GPS como mediante algoritmos de trilateración.

En la tabla 5.10 se presenta el TTFF medio para cada uno de los métodos, tanto en escenarios interiores como exteriores en una zona urbana:

Método	Situación	TTFF (ms)
Google Red	interiores	33195
GPS	exteriores	23227
GSM	interiores	121380
GSM	exteriores	109788

Tabla 5.10: TTFF en interiores y exteriores

Tal y como se exponía en el capítulo 4 existen una serie de factores que provocan que la latencia de la localización basada en métodos de trilateración es superior a la de GPS. Como es esperable, tanto en el caso de Google Red

como en el de GSM, la latencia es mayor cuando nos encontramos en el interior de un edificio debido a que la cobertura se ve atenuada y se tarda más tiempo en establecer una comunicación.

En una zona rural solo se pudo realizar la medida en exteriores ya que en interiores no se disponía de cobertura. En siguiente cuadro 5.11 muestra un resumen de los tiempos:

Método	Situación	TTFF (ms)
GPS	exteriores	14814
GSM	exteriores	14814

Tabla 5.11: TTFF en exteriores en Villafranca de la Sierra

En este caso, se puede observar que el tiempo es el mismo ya que en el caso de GSM no se realizaron peticiones HTTP al API de Google puesto que no disponíamos de información de las estaciones vecinas y asumíamos la posición de la estación a la que estábamos conectados como la posición del usuario, y por lo tanto, la latencia de GSM en escenarios rurales es menor que en casos urbanos, mientras que la de GPS es mayor ya que el dispositivo en zonas rurales tarda más en localizarnos y coincide con el de GSM puesto que inicialmente asume como posición del usuario la localización de la estación a la que estamos conectados y más tarde termina localizándonos correctamente.

Capítulo 6

Presupuesto

Este capítulo muestra un desglose de los costes del proyecto, tanto materiales como personales con el objetivo de presentar un coste total aproximado.

1. Autor:
   Débora Gómez Bertoli

2. Departamento:
   Ingeniería Telemática

3. Descripción del proyecto:

Título:	Estudio, implementación y análisis de métodos de trilateración para la localización de usuarios desde sus terminales móviles
Duración (meses):	12
Tasa de costes indirectos:	20 %

4. Presupuesto total del proyecto (valores en Euros):
   22.162 Euros

5. Desglose presupuestario (costes directos):

PERSONAL

Apellidos, Nombre	N.I.F.	Categoría	Dedicación (hombres mes)	Coste hombre mes	Coste (Euros)	Firma
Gómez Bertoli, Débora	47462557X	Ingeniero Junior	5	2.694,33	13.471,95	
Rodríguez Carrión, Alicia	49019028W	Ingeniero Senior	1	4.289,54	4.289,54	
		Hombres mes	6	**Total**	17.761,49	

EQUIPOS					
Descripción	Coste (Euros)	% uso dedicado al proyecto	Dedicación (meses)	Período depreciación	Coste imputable
HTC Desire	369	50	10	60	30,75
Portátil Dell Inspiron	600	100	12	60	120,00
TomTom one	129,95	50	6	60	6,50
				Total	157,25

SUBCONTRATACIÓN DE TAREAS		
Descripción	Empresa	Coste imputable
-	-	-
	Total	-

OTROS COSTES DIRECTOS DEL PROYECTO		
Descripción	Empresa	Coste imputable
Viaje a Villafranca de la Sierra	Departamento Ingeniería Telemática	50,00
Licencia MATLAB	Departamento Ingeniería Telemática	500,00
	Total	550,00

6. Resumen de costes:

Costes directos	
Personal	17.761
Equipos	157
Subcontratación de tareas	0
Costes de funcionamiento	550
Costes indirectos	3.694
Total	22.162

El presupuesto total de este proyecto asciende a la cantidad de 22.162 EUROS.

Leganés a 16 de Julio de 2012

El ingeniero proyectista

Fdo. Débora Gómez Bertoli

Capítulo 7

Historia del proyecto

Este capítulo tiene como objetivo describir el ciclo de vida de este proyecto de fin de carrera explicando la duración de cada una de sus fases y el conjunto de tareas desempeñadas.

Este proyecto puede dividirse principalmente en las siguientes fases:

- Investigación (01/07/2011 - 28/07/2011).

  Durante este perído se investigó sobre las aplicaciones y los servicios existentes basados en localización. Su principal objetivo era definir las propiedas de una aplicación diferente a las existentes y seleccionar aquellas técnicas de posicionamiento aplicables en un dispositivo móvil. Varias de las técnicas fueron descartadas puesto que debíamos conocer características de las estaciones emisoras como el ángulo de apertura o el nivel de potencia transmitida, dejándonos como única opción el posicionamiento mediante el identificador de celda y su combinación con métodos de trilateración. Una vez escogida la técnica de localización teníamos que estudiar la viabilidad de implementarlo en Android, por lo tanto, se hizo una valoración de las distintas APIs de Android para llevar a cabo esta función. El número de horas de dedicación fueron 57h.

- Desarrollo de la aplicación (18/07/2011 - 29/08/2011).

  Seleccionados los métodos de posicionamiento, se implementó en una primera subfase la localización mediante GPS y mediante GSM utilizando solo el identificador de celda junto con la escritura en fichero de cada una de las localizaciones y la información referente a las estaciones base vecinas y a las estaciones WiFi. En una segunda subfase y tras conocer detalladamente las ecuaciones referentes a los cuatro centros del triángulo (incentro, centroide, circuncentro y ortocentro), se procedió

a implementar cada uno de estos algoritmos y a presentar estas posiciones en la vista principal de la aplicación. Finalmente se estructuró el formato de los archivos y se definió la información a recoger por la aplicación. El número de horas dedicadas fueron 90h.

- Toma de medidas (31/08/2011 - 31/05/2012).

  Durante esta fase, se tomaron medidas en distintas localizaciones (Leganés y Villafranca de la Sierra) tanto en interiores como en exteriores con el objetivo de evaluar la calidad y la precisión de nuestra aplicación frente a la posición obtenida mediante Google (GPS o red). La principal dificultad que se afrontó en este período fue el clima para el caso de la toma de medidas en exteriores. El número de horas de dedicación fueron 393h.

  Como se refleja en la figura 7.1 esta tarea tiene una duración elevada puesto que la realización de todas las pruebas se fue coordinando con la fase de análisis de resultados. Además, durante esta fase hubo una parada de tres meses en los que no se llevó a cabo ninguna tarea, principal motivo de que esta tarea se demorara tanto tiempo.

- Análisis de resultados (31/08/2011 - 01/06/2012).

  El objetivo de esta fase fue sacar conclusiones y valoraciones de cada una de las características y funcionalidades de la aplicación Android y estudiar su viabilidad comercial. En este período no se encontraron dificultades puesto que el programa empleado para realizar el análisis (MATLAB) cumplía con todas las especificaciones requeridas para realizar la evaluación de la aplicación.

  Puesto que esta fase se realizó en paralelo con la toma de medidas, también sufrió una parada de tres meses durante su ejecución. La dedicación en términos de horas fue de 393h.

- Documentación (01/07/2011 - 18/06/2012).

  Como se puede apreciar en el gráfico Gantt del proyecto 7.2 es una tarea que se ha realizado en paralelo con el resto puesto que a la vez que se avanzaba, se iba documentando para agilizar la carga de trabajo que supone el desarrollo de este proyecto. Por lo tanto el número de horas es de 555h.

Esta figura 7.1 muestra detalladamente la duración de cada una de las fases del proyecto y de las tareas que las componen.

Esta figura 7.2 refleja en un diagrama Gantt la duración del proyecto.

Nombre de la tarea	*Duración*	*Fecha de inicio*	*Fecha de fin*
Investigación	57 hrs	Fri 01/07/11	Thu 28/07/11
Servicios basados en localización	12 hrs	Fri 01/07/11	Wed 06/07/11
Algoritmos de trilateración	48 hrs	Thu 07/07/11	Thu 28/07/11
Plataforma Android	15 hrs	Fri 08/07/11	Fri 15/07/11
API localización GSM	6 hrs	Fri 08/07/11	Mon 11/07/11
Geolocation Google API	6 hrs	Thu 14/07/11	Fri 15/07/11
Gestión de información	6 hrs	Thu 14/07/11	Fri 15/07/11
API localización GPS	6 hrs	Tue 12/07/11	Wed 13/07/11
API WiFi	6 hrs	Tue 12/07/11	Wed 13/07/11
Documentación	555 hrs	Fri 01/07/11	Mon 18/06/12
Introducción y objetivos	3 hrs	Fri 01/07/11	Fri 01/07/11
Estado del arte	48 hrs	Thu 07/07/11	Fri 29/07/11
Técnicas de posicionamiento	3 hrs	Thu 07/07/11	Thu 07/07/11
Métodos de trilateración	3 hrs	Fri 29/07/11	Fri 29/07/11
Plataforma Android	3 hrs	Mon 18/07/11	Mon 18/07/11
Desarrollo de DroidTriangulation	24 hrs	Fri 23/03/12	Wed 04/04/12
Introducción	18 hrs	Fri 23/03/12	Mon 02/04/12
Diagrama de clases	3 hrs	Mon 02/04/12	Mon 02/04/12
Interfaz gráfica	3 hrs	Tue 03/04/12	Tue 03/04/12
Funcionalidades	3 hrs	Wed 04/04/12	Wed 04/04/12
Entorno de Pruebas	45 hrs	Thu 05/04/12	Thu 12/04/12
Dispositivos empleados	3 hrs	Thu 05/04/12	Thu 05/04/12
Localizaciones	6 hrs	Thu 05/04/12	Fri 06/04/12
Medidas	18 hrs	Thu 05/04/12	Thu 12/04/12
Análisis de resultados	21 hrs	Fri 01/06/12	Mon 11/06/12
Historia del proyecto	9 hrs	Tue 12/06/12	Thu 14/06/12
Presupuesto	3 hrs	Fri 15/06/12	Fri 15/06/12
Conclusiones y trabajos futuros	3 hrs	Mon 18/06/12	Mon 18/06/12
Desarrollo de la Aplicación Android	90 hrs	Mon 18/07/11	Mon 29/08/11
Localización por GPS	9 hrs	Mon 18/07/11	Wed 20/07/11
Localización CID por GSM	9 hrs	Mon 18/07/11	Wed 20/07/11
Interfaz gráfica	6 hrs	Thu 21/07/11	Mon 25/07/11
Vista principal	9 hrs	Thu 21/07/11	Mon 25/07/11
Preferencias	3 hrs	Thu 21/07/11	Thu 21/07/11
Gestión de la información	6 hrs	Thu 21/07/11	Fri 22/07/11
Algoritmos de trilateración	66 hrs	Fri 29/07/11	Mon 29/08/11
Toma de medidas	393 hrs	Wed 31/08/11	Thu 31/05/12
Desarrollo programas análisis MATLAB	393 hrs	Wed 31/08/11	Fri 01/06/12

Figura 7.1: Lista de tareas y fases del proyecto

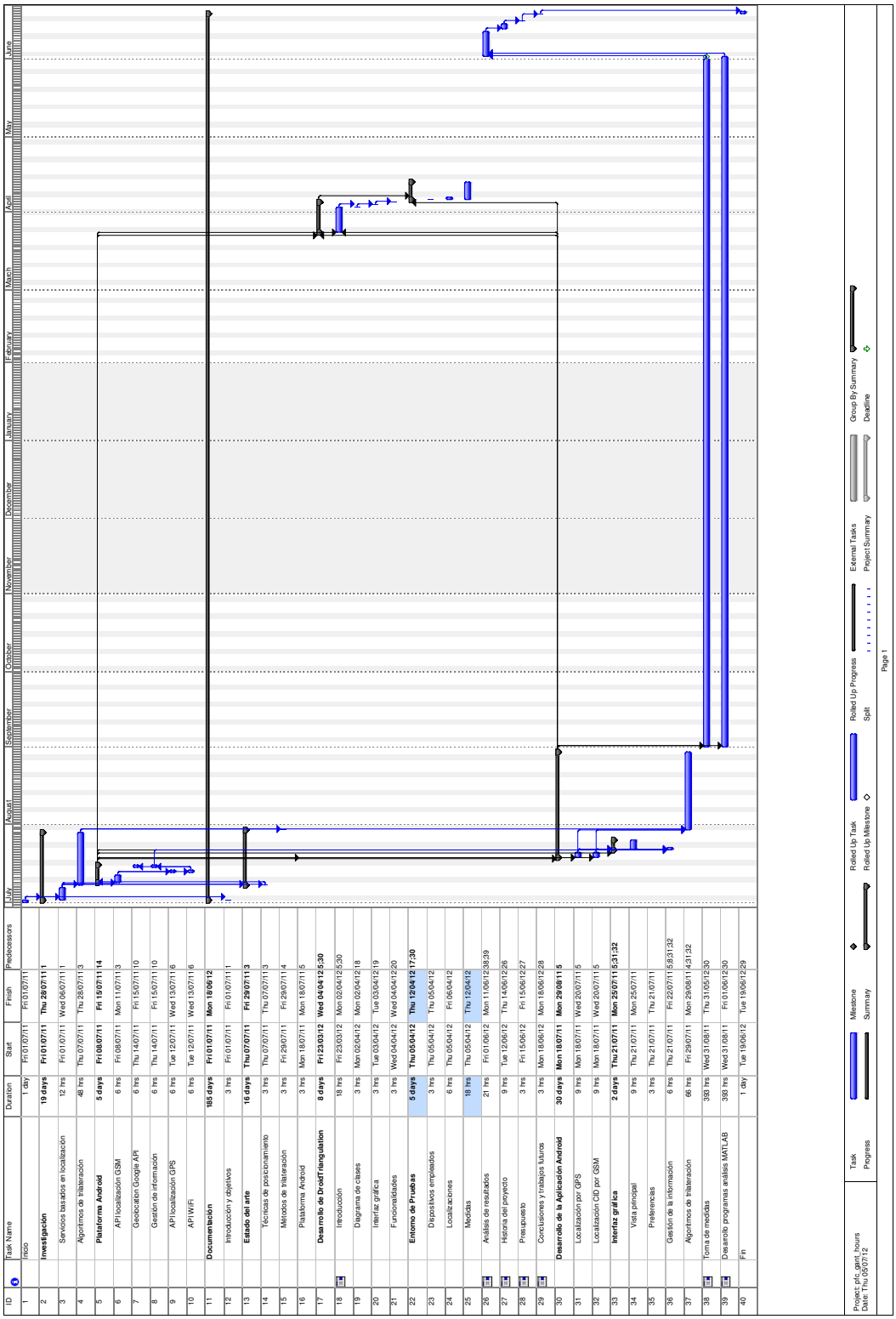


Figura 7.2: Diagrama Gantt del proyecto

Capítulo 8

Conclusiones y trabajos futuros

8.1. Conclusiones

Tal y como se ha ido planteando durante toda la memoria el obejtivo de esta aplicación para dispositivos con sistema operativo Android está orientado a estudiar la viabilidad de calcular la posición del usuario utilizando algoritmos de trilateración, partiendo de las tres estaciones base de las cuales obtenemos un mayor nivel de potencia, respecto al uso del GPS como medio de localización.

Una vez finalizadas cada una de las fases de este proyecto, podemos realizar un balance y crítica sobre los resultados obtenidos a lo largo de la vida de dicho proyecto.

- Estudio se los servicios de localización existentes y de algoritmos de triangulación aplicables.

  Antes de empezar a desarrollar la aplicación debíamos tener conocimiento de las posibles técnicas de localización existentes en el mercado actual y estudiar la viabilidad de si podrían ser aplicables sobre un dispositivo móvil, entre las cuales encontramos: A-GPS, CID o TA.

  Por otra parte, se estudió la posibilidad de calcular el centro de un triángulo empleando para ello como coordenadas baricéntricas la latitud y la longitud.

- Estudio de las principales características de Android.

  Android hace uso de un lenguaje muy extendido actualmente como lo es Java, y proporciona una serie de ejemplos en el propio SDK que son de gran ayuda para los programadores principiantes. Además, se trata de una plataforma libre dotada de una licencia Apache que permite

la distribución de aplicaciones a través de un mercado conocido como *Google Play*.

En lo referente a la estructura de una aplicación Android hemos visto en el capítulo 2 se compone de la interfaz, de una capa lógica y funcional, y permite la gestión de eventos nativos de la plataforma y el acceso a recursos del dispositivo, tales como GPS, WiFi, etc.

Finalmente, la documentación de apoyo para los desarrolladores y las publicaciones de artículos informativos y aclarativos sobre parte de las funcionalidades de Android, hacen que sea más comprensible y fácil programar en este lenguaje.

- Desarrollar una aplicación de localización en Android.

  DroidTriangulation es una aplicación que permite conocer la posición del usuario haciendo uso del GPS o de métodos de trilateración aplicados a partir de las coordenatas de tres estaciones base conocidas.

  La estructura de la aplicación está dividida por capas para separar de forma adecuada la parte funcional, la parte lógica y la parte de interfaz para que sea más comprensible de cara a los futuros programadores que tomen la aplicación.

  Se trata de una aplicación muy completa que hace uso de las APIs más importantes de esta plataforma, y que por tanto, también sirve como base de ejemplo de aplicaciones basadas en la localización del usuario.

- Análisis de los resultados de la aplicación Android.

  Como hemos visto a lo largo del capítulo 5 la aplicación presenta una serie de inconvenientes frente a los servicios de localización existentes que hacen uso, únicamente, de la tecnología GPS. Entre estos inconvenientes, destaca el hecho de que no sea tan preciso como es el GPS, sin embargo, cuando no tenemos señal GPS en interiores y el móvil reciba señal de la red, podremos utilizar la aplicación para tener una idea aproximada de donde nos situamos.

  Por otra parte, también se ha confirmado que el tiempo que tarda la aplicación en obtener la primera posición del usuario es superior para el caso del uso de algoritmos de trilateración debido a que se deben realizar tres peticiones HTTP para obtener la latitud y la longitud de las tres estaciones base implicadas en el cálculo.

  No obstante, estos inconvenientes pueden ser compensados por el hecho de que el consumo de batería de la aplicación es significativamente menor cuando el usuario sólo tiene activada la opción de localización

mediante GSM. se trata de un punto fuerte ya que la principal desventaja de los servicios basados en localización que utilizan GPS realizan un consumo elevado de batería, lo cual es bastante crítico ya que los *smartphones*, o teléfonos inteligentes, actualmente consumen mucha potencia y el hecho de utilizar el GPS del teléfono móvil provoca que la batería de estos teléfonos llegue a durar, en la mayoría de los casos, menos de un día.

8.2. Trabajos futuros

Nuestra aplicación, por el momento, calcula una posición en función de las tres estaciones base de las cuales obtenemos mayor potencia haciendo uso de métodos de trilateración. Este cálculo puede ser mejorado aplicando modelos de propagación como puede ser el modelo de Okumura-Hata para las zonas urbanas [25] [19]. Sin embargo, uno de los modelos que se plantea con el uso de modelos de propagación es que es necesario conocer la potencia transmitida. Actualmente, Google no nos proporciona esta información y como aproximación se podría considerar que todas las estaciones base transmiten con la misma potencia, y por tanto, trabajar con potencias relativas en las fórmulas de modelos de propagación.

En la misma línea de investigación, el gobierno español ha puesto a disposición una página web `http://geoportal.mityc.es/visorCartografico/index.jsp` en la que se puede consultar la situación de estaciones base móviles e información sobre la potencia recibida a diferentes ditancias. De modo que, podríamos considerar como potencia transmitida la información referente a la potencia recibida para la distancia más corta. La figura 8.1 muestra un ejemplo de la información proporcionada por esta página web.

Una posible mejora sería utilizar las herramientas de desarrollo nativo para Android, que nos permitiría hacer los cálculos de los algoritmos de trilateración a bajo nivel utilizando el lenguaje de programación C, y por tanto, estos algoritmos serían portables para ser ejecutados en otras plataformas como puede ser iOS. Para ver más información sobre esta herramienta visitar `http://developer.android.com/sdk/ndk/overview.html`

En el capítulo 4 se comentaba que al querer medir la latencia tanto de la localización por GPS como por GSM, no se ha proporcionado un mecanismo de caché para guardar la última posición obtenida mediante algoritmos de trilateración. La inclusión de una caché implicaría una reducción del tiempo de espera para la primera localización del usuario cuando la aplicación se hubiese utilizado en ocasiones anteriores, y un ahorro de energía porque guardaría los datos de las peticiones HTTP y no tendría que repetirlas.

El marco de desarrollo de este proyecto puede llevarse a la localización en interiores, por ello una de las funcionalidades accesibles desde el menú de la aplicación es realizar un barrido de las estaciones WiFi que nos rodean. Una posible línea futura de desarrollo sería aplicar técnicas de *fingerprinting*, de tal modo que, realizando una fase previa para crear un mapa del interior de los edificios que contenga las estaciones WiFi, podamos en tiempo real localizarnos por cercanía a alguna de estas estaciones, por ejemplo.

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

Niveles de exposición

LOCALIZACIÓN:

Código del emplazamiento:00288
Provincia:MADRID
Ciudad:LEGANÉS
Ubicación:CL VIRGEN DEL CAMINO, 1
Coordenadas:03W4609.83 40N1947.91

CARACTERÍSTICAS TÉCNICAS

Operador	Código estación	Sistema	Banda	Valor Permitido (µW/cm²)
VODAFONE ESPAÑA, S.A.	0357MX	UMTS	1905-1910; 1950-1965; 2140-2155 MHz	1000.00µW/cm²
VODAFONE ESPAÑA, S.A.	c-357	DCS	1825.1-1845.1 MHz	900.00µW/cm²
VODAFONE ESPAÑA, S.A.	i-157	GSM	948.9-959.9 MHz	450.00µW/cm²

NIVELES MEDIDOS EN EL ENTORNO

Distancia (m)	Acimut (º)	Valor Medido (µW/cm²)
25.0m	10.0º	0.03µW/cm²
31.0m	20.0º	0.13µW/cm²
34.5m	245.0º	0.04µW/cm²
40.0m	0.0º	0.23µW/cm²
46.0m	120.0º	0.07µW/cm²
55.0m	190.0º	<0.01µW/cm²
65.0m	45.0º	0.13µW/cm²
65.0m	180.0º	<0.01µW/cm²
66.0m	255.0º	0.12µW/cm²
20.0m	165.0º	0.03µW/cm²
18.5m	315.0º	0.06µW/cm²

Página principal

Aviso legal | Protección de Datos Personales

Gobierno de España. Ministerio de Industria, Energía y Turismo
Pº de la Castellana 160, C.P.28046Madrid.
Teléfono 902 446 006

Figura 8.1: Resumen de características de una estación base

Referencias

[1] Android developers. http://developer.android.com/develop/index.html (último acceso, 10 septiembre 2011).

[2] Android security. http://developer.android.com/guide/topics/security/security.html (último acceso, 15 julio 2011).

[3] Avoiding memory leaks. http://android-developers.blogspot.com/2009/01/avoiding-memory-leaks.html (último acceso, 15 julio 2011).

[4] Data storage. http://developer.android.com/guide/topics/data/data-storage.html (último acceso, 20 agosto 2011).

[5] Geolocation API. https://developers.google.com/gears/l, (último acceso, 2 noviembre 2012).

[6] Google maps API. http://developer.android.com/guide/topics/location/index.html, último acceso 2 noviembre 2011.

[7] Obtaining user location. http://developer.android.com/guide/topics/location/obtaining-user-location.html, (último acceso, 15 octubre 2011).

[8] Painless threading. http://developer.android.com/resources/articles/painless-threading.html (último acceso, 15 julio 2011).

[9] Tracking memory allocations. http://android-developers.blogspot.com/2009/02/track-memory-allocations.html (último acceso, 15 julio 2011).

[10] Paul A. Zandbergen. Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning. *Transactions in GIS*, 13:5–25, Jun 2009.

[11] Muhammad Aatique. Evaluation of TDOA techniques for position location in CDMA systems. Master's thesis, Faculty of the Virginia Polytechnic Institute and State University, Virginia, USA, 1997.

[12] Abraham Albert Ungar. *Barycentric Calculus in Euclidean and Hyperbolic Geometry: A comparative introduction*. World Scientific Publishing Co. Pte. Ltd., 2010.

[13] Mikkel Baun Kjærgaard, Henrik Blunck, Torben Godsk, Thomas Toftkjær, Dan Lund Christensen, and Kaj Grønbæk. Indoor positioning using GPS revisited. *ACM*, pages 38–56, 2011.

[14] Alan Bensky. *Wireless Positioning: Technologies and Applications*. Artech House, Inc, 2008.

[15] Tim Bray. Processing ordered broadcast. `http://android-developers.blogspot.com/2011/01/processing-ordered-broadcasts.html` (último acceso, 30 julio 2011), Enero 2011.

[16] Christopher Drane, Malcolm Macnaughtan, and Craig Scott. Positioning for GSM telephones. *IEEE Communications Magazine*, 36:46–54, April 1998.

[17] Bohdan Dulya. GSM positioning. `http://www.ks.uni-freiburg.de/download/papers/lbssemWS08/Folien/Bohdan_Dulya_GSM_Slides.pdf`, Febrero 2009.

[18] Kristina Dunbar. Triangle centers. `http://jwilson.coe.uga.edu/emat6680/dunbar/assignment4/assignment4_kd.htm`.

[19] M. Hata. Empirical formula for propagation loss in land mobile radio services, 1980.

[20] Jeffrey Hightower and Gaetano Borriello. Location sensing technologies and applications. *JISC Techwatch Report*, Agosto 2001.

[21] Deni Huber. Background positioning for mobile devices - Android vs iPhone. `http://www.snet.tu-berlin.de/fileadmin/fg220/courses/WS1011/snet-project/background-positioning_huber.pdf`.

[22] Clark Kimberling. Encyclopedia of triangle centers - ETC. `http://faculty.evansville.edu/ck6/encyclopedia/ETC.html` (último acceso, 25 octubre 2011).

[23] Goran M. Djuknic and Robert E. Richton. Geolocation and assisted GPS. *ACM*, pages 123–125, Feb 2001.

[24] Jason Nieh. Mobile Computing with Android and iPhone: Location Based Services. `http://www.cs.columbia.edu/~nieh/teaching/e6998/lectures/lecture9.pdf`, 2008.

[25] Y. M. Okumura, E. Ohmori, T. Kawano, and K. Fukuda. Field strength and its variability in the VHF and UHF land mobile radio service. *Elec. Comm. Lab*, 9–10:825–873, 1968.

[26] Yilin Zhao. Standarization of mobile phone positioning for 3G systems. *IEEE Communications Magazine*, 40:108–116, Jul 2002.

Apéndice A

Manual de instalación

Para instalar la aplicación sobre un dispositivo Android debemos conectar el teléfono mediante USB al ordenador. Para ello hay que activar la opción "Unidad de disco" en el menú de tipo de conexión, tal y como se muestra en la figura A.1.

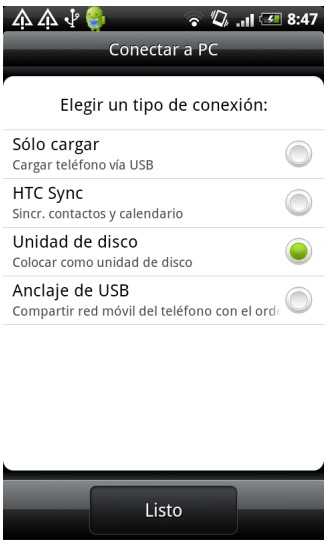


Figura A.1: Menú de tipo de conexiones en el móvil

Copiamos el archivo `DroidTriangulation.apk` a una carpeta de la tarjeta SD del móvil y lo desconectamos del ordenador, o bien desactivamos la opción "Unidad de disco". A continuación, usando un programa explorador de archivos, normalmente todos los teléfonos vienen con uno preinstalado, navegamos hacia la carpeta donde hayamos copiado el archivo y hacemos click sobre el ejecutable. Nos aparecerá el menú de instalación del teléfono tal y como se presenta en la figura A.2

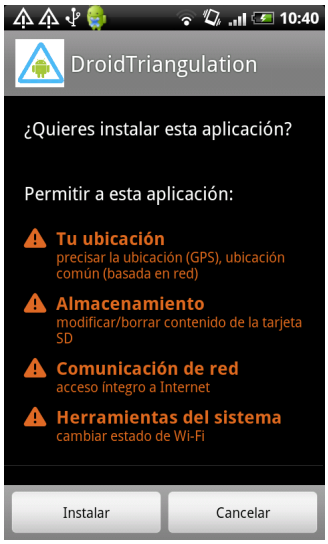


Figura A.2: Menú de instalación en el móvil

Si la instalación se realizó correctamente aparecerá una pantalla con la siguiente información A.3:

Ya se puede proceder a utilizar la aplicación.

Figura A.3: Menú de instalación finalizada en el móvil

Apéndice B

Manual de usuario

Antes de empezar a utilizar la aplicación debemos asegurarnos de que el GPS se encuentra activado y la conexión de datos también. Esto se puede hacer desde el menú de ajustes del terminal móvil.

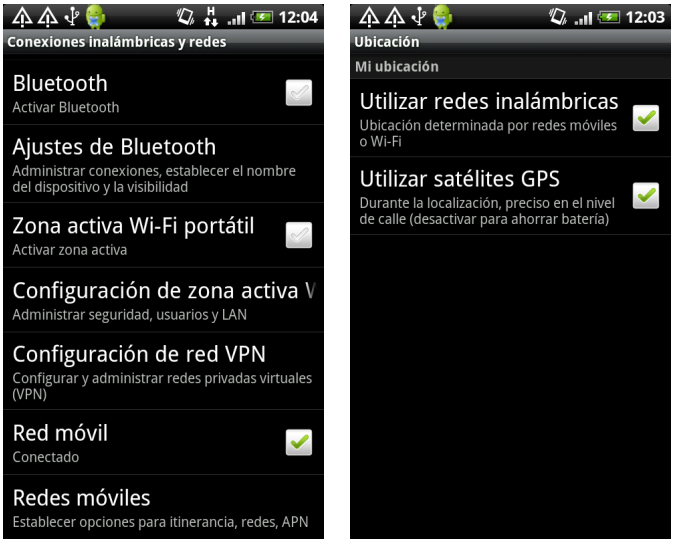


Figura B.1: Pantallas de configuración de GPS y conexión de datos en el móvil

Para poder ejecutar la aplicación accedemos al menú de aplicaciones del teléfono móvil y pulsamos sobre el icono de la aplicación *DroidTriangulation* B.2.

Figura B.2: Menú de aplicaciones en el móvil

Lo primero que visualizaremos será una pantalla de inicio que se muestra mientras la aplicación hace una serie de comprobaciones y a continuación se mostrará la vista principal de la aplicación B.3:

Figura B.3: Pantalla de inicio de la aplicación

La primera vez que se ejecuta la aplicación nos intentará localizar tanto por GSM como por GPS. Para cambiar estas opciones, pulsamos sobre la tecla "Menu" del terminal móvil, y accedemos a las preferencias de la aplicación.

En la primera pestaña podemos elegir el tipo de localización a emplear por la aplicación, tal y como se muestra en la figura B.4:

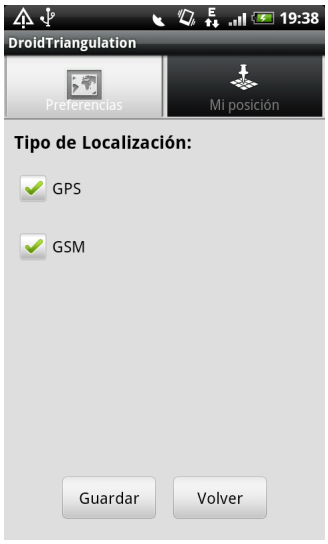


Figura B.4: Menú de preferencias en el móvil

En la segunda pestaña podemos consultar la posición correspondiente a cada uno de los centros como se muestra en la figura B.5

Figura B.5: Menú de preferencias con localizaciones en el móvil

Para recoger medidas de las estaciones WiFi que nos rodean, debemos activar este tipo de conexión desde el menú ajustes del teléfono B.6:

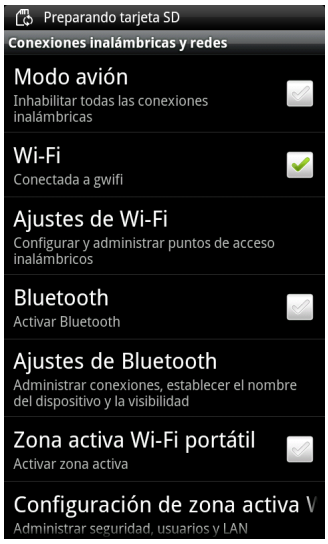


Figura B.6: Menú de conexiones inalámbricas en el móvil

Apéndice C

Configuración Google Maps

Antes de registrarnos para poder utilizar este servicio debemos conocer una serie de aspectos:

- Debemos tener una cuenta de Google para poder obtener la clave puesto que dependerá de nuestro usuario.
- No hay límites en el número de vistas que queramos generar usando esta API.
- Además, esta API no contiene publicidad, pero en el caso de que Google decida cambiarla, seremos notificados.
- Nuestro servicio debe estar disponible gratuitamente para todo el mundo.

Para poder mostrar una vista en nuestra aplicación de *Google Maps* necesitamos obtener una clave que nos de permisos.

Para obtener la clave tenemos que utilizar una herramienta de java llamada `keytool`.

- Localizar la dirección donde se encuentra dicha herramienta dentro de nuestro ordenador. Para ello, podemos utilizar Eclipse donde en la opción Window - Preferences - Android - Build, en la opción `Default debug keystore` encontraremos la dirección.
- Abrimos una ventana de línea de comandos, vamos a dicha dirección y escribimos:

```
1 keytool -list -alias androiddebugkey -keystore (path to
    keystore)\debug.keystore -storepass android -keypass
    android
```

- Nos deberá mostrar un mensaje como el siguiente:

```
1 androiddebugkey, Sep 1, 2011, PrivateKeyEntry, Certificate
    fingerprint (MD5): XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:
    XX:XX:XX:XX
```

Finalmente, debemos ir a la siguiente página `http://code.google.com/intl/ja/android/maps-api-signup.html`, aceptar los términos y condiciones, y copiar la clave generada por linea de comandos.

Si todo ha ido correctamente, nos aparecerá una página con el siguiente mensaje: *Gracias por suscribirte a la clave del API de Android Maps. Tu clase es: (your key)*

Esta clave será la que se especifique en el elemento XML `com.google.android.maps.MapView`, concretamente en el atributo `apiKey`.